UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

SERVIDOR DE MAPAS PARA PUBLICACIÓN DE RUTAS SENDERISTAS E INTERFAZ WEB CLIENTE

Realizado por

JOSÉ ROLLAND LÓPEZ DE COCA

Dirigido por

SEBASTIÁN CASTILLO CARRIÓN

JESÚS VÍAS MARTÍNEZ

Director Académico

JOSÉ DEL CAMPO ÁVILA

Departamento

LENGUAJE Y CIENCIAS DE LA COMPUTACIÓN

MÁLAGA, Julio de 2011

A mi familia, a mis amigos
y a todos los que me han ayudado a llegar hasta aquí.

Índice

BLOQUE 1: INTRODUCCIÓN AL PROYECTO

CAPÍTULO

1 Introducción

La información geográfica y el uso de herramientas y sistemas que permitan hacer uso de ella para extraer nueva información y ponerla a disposición de la sociedad es una tarea que engloba diferentes áreas de conocimiento, como la Geografía o la Informática. Desde el departamento de Geografía de la Universidad de Málaga se está trabajando en la creación de nuevos contenidos y en su difusión en el ámbito de diferentes proyectos de investigación, como son:

- el proyecto de excelencia P07-HUM-03049: "Desarrollo metodológico sobre la evaluación de la capacidad para usos recreativos de espacios protegidos" financiado por la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía.

- el proyecto del Plan Nacional de Investigación SEJ2007-67690: "Desarrollo metodológico sobre la evaluación de la capacidad para usos recreativos de espacios protegidos" financiado por la Dirección General de Investigación del Ministerio de Ciencia e Innovación.

Los anteriores proyectos pretenden desarrollar métodos de evaluación de la aptitud y de la capacidad de carga del territorio con relación al conjunto de las actividades recreativas ligadas a los caminos o senderos, que son las contempladas más frecuentemente en las figuras de planeamiento (Plan de ordenación de los recursos naturales y Plan de Uso Público) de estos espacios.

Con esta perspectiva se aborda, entre otros objetivos, la evaluación de los caminos-senderos para el disfrute recreativo, con atención a sus caracteres intrínsecos, como soporte de las actividades y dando cabida a la valoración del entorno y el paisaje.

Para ello proponen un sistema que combine técnicas multicriterio y los modelos de capacidad de acogida, configuradas como una aplicación en un Sistema de Información Geográfica (SIG), que no es más que un sistema que auna tanto capacidades de cálculo geográfico (el llamado procesamiento geográfico) como de manipulación gráfica y de imágenes, consulta y gestión de bases de datos. Esta aplicación será presentada a usuarios finales para poner a prueba el sistema, de forma que pueda servir de modelo para procesos de

evaluación de esta índole [1, 2, 3]. Concretamente se trabajará con los datos de una zona piloto, el Parque Natural de Sierra de las Nieves, aunque la aplicación dejará abierta la posibilidad de incorporar posteriormente datos de otras zonas.

El desarrollo de herramientas y aplicaciones capaces de manejar datos geográficos, junto con el auge de Internet, y sin duda apoyado en el proceso globalizador que estamos viviendo en todos los ámbitos, son los responsables de fomentar extraordinariamente el intercambio de información espacial, y ha permitido que esta información, a través de aplicaciones web, está disponible para su uso en cualquier lugar con acceso a Internet.

El conjunto de tecnologías que facilitan la integración, la disponibilidad y el acceso a la información espacial en Internet se denomina Infraestructura de Datos Espaciales (IDE). Existen varias definiciones sobre qué es una IDE, pero todas destacan los siguientes puntos: una IDE debe estar formada por conjuntos de datos espaciales, servicios de datos geográficos espaciales y metadatos (que serán el índice que describa los datos y los servicios), junto con las tecnologías de red necesarias para proporcionar acceso a esos datos [4, 5].

Otro aspecto interesante es la capacidad de comunicación entre diferentes IDEs, de forma que cada una de ellas actúe como un nodo IDE dentro de una red más amplia de IDEs interconectadas. Esto implica poner en común acuerdos y políticas institucionales, especificaciones normalizadas y uso de estándares que ayuden al intercambio de información en estos sistemas informáticos distribuidos. Una IDE puede implementarse en una empresa, un centro de investigación, un organismo oficial, como ayuda para la gestión de su propia información espacial, y también puede implantarse como servicio público creado ex-profeso y/o enlazando otras IDEs.

La relación entre los proyectos anteriores y el presente proyecto fin de carrera surge a partir de la elaboración de un nodo IDE y una aplicación web que permita poner en valor los recursos de una zona (en este caso la Sierra de las Nieves) y muestre a cualquier usuario de ese recurso la potencialidad del medio siempre bajo la perspectiva del desarrollo sostenible.

De esta forma, se proporcionará al usuario, mediante el uso de dicha aplicación web, la ruta más afín a sus preferencias dentro de las posibilidades contempladas en el Sistema de Información Geográfica (SIG). Para ello, la herramienta deberá recoger diferentes características (capacidades físicas, intereses, etc.) del usuario y las usará conjuntamente con parámetros calculados y con criterios específicos para, haciendo uso de métodos de búsqueda previamente implementados, ofrecer las rutas más apropiadas.

CAPÍTULO

2 Definición del proyecto

En el presente capítulo se definirá de forma detallada el proyecto que se pretende abordar, así como todo lo que rodea al mismo.

2.1. Descripción del proyecto

Tal y como se ha expuesto en el capítulo anterior, el objetivo es diseñar e implementar una Infraestructura de Datos Espaciales (IDE) y una aplicación web que haga de interfaz entre el usuario y la IDE y facilite al cliente el cálculo de rutas senderistas de la Sierra de las Nieves, utilizando distintas tecnologías y lenguajes de programación enfocados hacia la Web. En la figura 2.1 se muestra todo esto a modo de esquema:

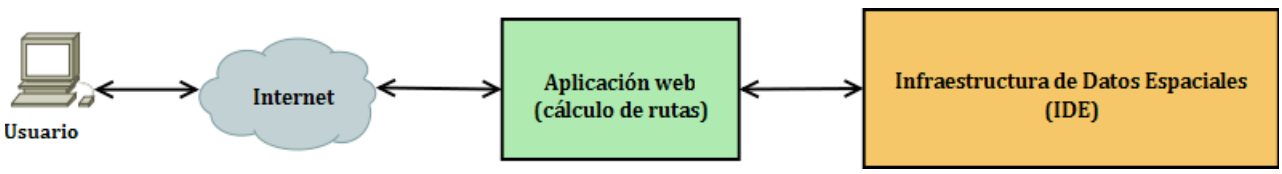

Figura 2.1. Esquema básico del proyecto.

De esta forma se pretende desarrollar una infraestructura que permita:

- Dar visibilidad en la Web a los datos geográficos espaciales (que llamaremos, abreviadamente, datos geoespaciales) recopilados por el Departamento de Geografía de la Universidad de Málaga.

• Automatizar el proceso del cálculo de perfiles y de rutas senderistas, que hasta el momento se hacía sobre el papel.

• Instalar una IDE con herramientas de software libre, dada la compatibilidad multiplataforma, libertad de uso y redistribución, libertad de modificación y mejora o el amplio soporte que suele mostrar este tipo de software, y por el que muchas administraciones están apostando.

• Diseñar el sistema de forma que permita posteriormente incluir información de cualquier otra zona, de forma que toda su funcionalidad se pueda adaptar a cualquier territorio más allá de la Sierra de las Nieves.

Con el sistema instalado y en funcionamiento, la aplicación proporcionará como características más destacadas:

• El usuario podrá buscar la ruta más afín a sus preferencias dentro de las posibilidades contempladas en la Infraestructura de Datos Espaciales (IDE).

• El sistema recogerá diferentes características (capacidades físicas, intereses, etc.) del usuario y las usará conjuntamente con parámetros calculados y con criterios específicos para, haciendo uso de métodos de búsqueda previamente implementados, ofrecer las rutas más apropiadas.

• El usuario visualizará las rutas encontradas por el sistema en un mapa interactivo, con información adicional de los senderos (longitud, tiempo estimado en recorrer la ruta, etc.).

• El usuario podrá registrarse en el sistema, y éste calculará su perfil en función de unos determinados parámetros.

• La aplicación gestionará los usuarios que hagan uso del sistema para poder realizar tareas más avanzadas que complementen la búsqueda de rutas de forma anónima (controlar rutas ya realizadas, recuperar información sobre capacidades físicas del usuario, etc.).

• Cualquier incidencia o evento inesperado que ocurra con el uso de la aplicación, será avisado (mediante un mensaje) y manejada por la misma.

• El usuario registrado podrá descargar las rutas ofrecidas para la aplicación, para su posterior uso en dispositivos específicos de localización geográfica (GPS).

• Se permitirá el acceso a usuarios no registrados, pero al no haber perfil no se guardarán ni se podrán descargar las rutas consultadas.

• La aplicación ofrecerá información general relativa a la zona de senderos, la Sierra de las Nieves, así como enlaces relacionados.

2.2. Entorno

El entorno se puede definir como una serie de propiedades relativas a un proyecto que pueden influir en el planteamiento y desarrollo del mismo, menoscabando en potencia al sistema software desarrollado para tal fin. Para analizar el entorno de la aplicación web que se pretende desarrollar, se puede diferenciar entre los siguiente tipos de entorno:

• *Entorno de programación*: para el desarrollo de la aplicación web se hará uso de los lenguajes de programación necesarios para desarrollar la aplicación web y actuar con la IDE.

• *Entorno software*: se refiere a los componentes software necesarios para que la aplicación pueda funcionar de forma correcta:

  ‣ Por la parte del cliente, como se trata de una aplicación web, tan sólo hará falta disponer de un navegador web que cumpla con los estándares de consorcio W3C y permita ejecutar *scripts*. En la actualidad la inmensa mayoría de navegadores web cumplen con estos requisitos.

  ‣ Por la parte del servidor, deberá disponer del software necesario para ejecutar la aplicación web y la IDE.

• *Entorno hardware*: este entorno está enfocado a las características físicas del sistema informático en el que correrá la aplicación:

  ‣ Por el lado del servidor, hará falta disponer de un sistema persistente que dé servicio en todo momento a la IDE y que sea capaz de atender todas las peticiones de los clientes que llegan a través de Internet. Dependiendo de la carga, número de usuarios habituales que visiten la web, harán falta más o menos prestaciones técnicas.

  ‣ Por el lado del cliente, un dispositivo informático con acceso a Internet y con un navegador web instalado.

• *Entorno de usuario*: la usabilidad, entendida como la facilidad de uso de una interfaz web (Jakob Nielsen [6]), tendrá un papel importante durante el desarrollo del proyecto, y el objetivo final es que los usuarios utilicen la aplicación de la forma más rápida, fácil, y amigable posible. Ello redundará en la eficiencia en la realización de la tarea y en una mayor satisfacción del usuario que la lleva a cabo.

2.3. Vida esperada

La vida esperada de una aplicación informática se puede definir como el tiempo estimado durante el cual puede ser útil. Esta estimación es difícil de calcular, pero aún más difícil es definir cuándo es útil y cuándo deja de serlo. No obstante, podemos aproximarnos especificando que esa utilidad puede referirse a la tecnología empleada en la aplicación, si es actual y funciona o si por el contrario está obsoleta y no es operativa.

Para una aplicación web, es difícil que su vida esperada termine puesto que las nuevas tecnologías y los navegadores que visualizan y ejecutan esas aplicaciones mantienen una gran retrocompatibilidad, así que en este punto podríamos extender la definición y añadir nuevas condiciones: su capacidad de adaptarse a las nuevas tecnologías y actualizaciones de lenguajes de programación que surjan, con objeto de poder actualizarla, facilitar su mantenimiento y añadirle nuevas funcionalidades. Por ejemplo, una aplicación web programada antes de la aparición de CSS, donde todo el aspecto visual había que escribirlo en HTML, hoy día habría terminado con su vida útil (tal y como se acaba de definir), y tendríamos que reprogramarla para poder hacer uso de las nueva funciones que trae CSS si queremos dotar a la aplicación web de una mayor riqueza visual.

No es recomendable diseñar la aplicación pensando en ningún navegador en concreto, ya que no se puede conocer cuál será la evolución de dicho navegador. Para que la aplicación llegue al mayor ámbito posible de usuarios, hay que tener presente el concepto de *cross-browser* (cuya traducción al español podría ser "multi-navegador"), cuya meta es la compatibilidad con todos los navegadores. Para lograrlo no sólo basta con seguir los estándares, sino que también hay que tener presente y abordar los casos excepcionales de navegadores populares que no siguen esos estándares, para así lograr la compatibilidad completa.

2.4 Ciclo de mantenimiento

La capacidad de una aplicación software para sufrir las modificaciones necesarias frente a nuevos requisitos que sean demandados por parte del usuario o del sistema, marcará su ciclo de mantenimiento. Esas nuevas circunstancias suelen ser de cinco tipos:

- Prevención: conjunto de modificaciones para detectar y corregir fallos latentes antes de que se conviertan en fallos operacionales.
- Perfeccionamiento: mejorar la funcionalidad de la aplicación, aumentar su eficiencia o añadir nuevas funcionalidades.
- Revisión: revisar los requisitos originales.
- Adaptación: adaptar la aplicación a un nuevo entorno tecnológico.
- Corrección: corregir los errores no detectados durante el desarrollo de la aplicación hasta ese momento.

Para favorecer el mantenimiento de una aplicación es imprescindible proporcionar una documentación detallada del código, y es importante tener en cuenta factores como la escalabilidad, que en el caso que nos ocupa es muy positiva, dado el diseño modular que ostenta, y ayuda a ahorrar tiempo y trabajo y evitar efectos dominó (esto es, que un cambio provoque cambios de comportamiento inesperados en otro lugar de la aplicación).

2.5 Ámbito

La aplicación web a desarrollar pretende complementar a otras existentes, como Wikiloc (www.wikiloc.com) o Bikemap (www.bikemap.net). Éstas están enfocadas hacia un contenido más social, donde es el usuario el que crea y comparte rutas con otros. Webs como Vía Michelin (www.viamichelin.es) ofrecen capacidades de enrutamiento, con búsquedas del camino más corto, el más rápido (priorizando el uso de autovías) o el más económico en cuanto a ahorro de combustible, en función del terreno, pero no adapta la ruta en función del vehículo que se va a usar o del motivo de ese viaje.

La aplicación web a implementar aspira a cubrir un ámbito nuevo, complementando a las webs existentes, y se centrará en ofrecer un sistema de inteligencia artificial de búsqueda de rutas, con un análisis de las aptitudes físicas y capacidades del usuario para que el sistema las ofrezca en función de eso.

2.6 Aspecto visual

El aspecto visual de un sitio web es fundamental para captar al usuario y ofrecer una navegación clara, intuitiva y a la vez agradable por sus páginas. Así pues, hay que centrar gran parte de los esfuerzos en dotar a la aplicación de un interfaz de usuario en esa dirección.

En capítulos posteriores se desarrollará el diseño e implementación de la interfaz de usuario para la aplicación web motivo de este proyecto.

2.7 Estandarización

Uno de los puntos clave en el desarrollo de una aplicación web tiene que ver con los estándares, dado que la aplicación va a ejecutarse en diferentes navegadores y la mejor forma de hacerlo correctamente en todos ellos es siguiendo los estándares. Sin embargo, no todos los navegadores cumplen los estándares, y se hace imprescindible añadir código específico para esos navegadores, y así conformar una aplicación *cross-browser*, es decir que sea compatible con cualquier navegador. Afortunadamente, en los últimos meses las nuevas versiones de los navegadores se están esforzando por acercase a lo que marca el consorcio W3C (World Wide Web Consortium, el organismo encargado de regular los estándares de la Web [7]), e incluso algunos casos clásicos de navegadores ligeramente alejados de los estándares (como el de Microsoft con Internet Explorer) se han puesto a la cabeza en el estricto cumplimiento de los nuevos estándares, relativos a HTML5.

Lo mismo ocurre con la construcción del nodo IDE, si queremos que por ejemplo pueda interactuar con otros nodos IDE. El consorcio OGC (Open Geospatial Consortium) es en este caso el encargado de la estandarización de los datos y servicios que involucran información geoespacial [8]. Así pues, para la IDE proyectada se usarán, siempre que sea posible, los estándares del OGC.

Como ya se ha comentado, para la parte de la programación web se siguen las directrices del consorcio W3C. También se aplicarán la mayor parte posible de características de accesibilidad web (mecanismos para favorecer el acceso a la web a personas con algún tipo de discapacidad, o a cualquier otra persona que se encuentre bajo circunstancias externas que dificulten su acceso a la información) contempladas por el W3C. En la página web del W3C se encuentra un validor tanto para HTML como para CSS, que como su propio nombre indica analiza si un documento HTML o CSS siguen los estándares de la W3C. Durante el desarrollo de la aplicación se usará el validador para comprobar que sigue los estándares.

Para definir estándares abiertos para datos y servicios geoespaciales y asegurar interoperabilidad entre los Sistemas de Información Geográfica dentro del contexto de la World Wide Web, en 1994 se creó el consorcio Open Geospatial Consortium (OGC), que reuniría en una sola organización anteriores intentos al respecto como la fundación OGF (Open GRASS Foundation) o el CERL estadounidense. Entre ellos se encuentra el estándar que define la visualización de mapas vía Internet, llamado Web Map Service (WMS).

Web Map Service (WMS) es un protocolo para visualizar mapas de datos geoespaciales de forma dinámica a partir de información geográfica en formato de imagen (como PNG, GIF o JPEG), o vectorial (SVG). Define tres operaciones: devolver metadatos del nivel de servicio; devolver un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos y devolver información de características particulares mostradas en el mapa (opcionales).

Además existe otra organización, llamada Open Source Geospatial Foundation (OSGeo), que promueve el uso y da soporte al software geoespacial de código abierto que cumple los estándares OGC [9].

2.8 Calidad y fiabilidad

Estos dos factores son muy importantes a la hora de desarrollar cualquier aplicación, con objeto de ganarse la confianza del usuario y asegurar que va a funcionar con garantías.

La calidad de una aplicación web se asocia a dos hechos diferenciados:

- Su disponibilidad permanente, es decir, que el servidor siempre esté disponible y no llegue a saturarse. Se trata de algo a posteriori y que no depende del desarrollador, pero sin duda hay que intentar lograr que la aplicación consuma la menor cantidad de recursos del servidor, esto es, optimizar su eficiencia e intentar en la medida de lo posible que sea el navegador del usuario el que

ejecute la mayoría de funciones (JavaScript), aprovechando el poder de computación del dispositivo del cliente.

- Que no se produzcan errores inesperados que bloqueen el sistema o la aplicación.

La fiabilidad, por su parte, hace referencia a la capacidad de la aplicación para proporcionar siempre datos reales y correctos, es decir, asegurar su correcto y óptimo funcionamiento. En esta aplicación la fiabilidad es un punto fundamental, ya que las rutas que le ofrezcamos deben estar referenciadas geográficamente de la forma más exacta posible, y cuando las cargue en su GPS transcurran por senderos que existen y se adecuan a lo que pidió.

2.9 Programa de tareas

El programa de tareas se define como las diferentes fases de desarrollo por las que pasa la aplicación, que se pueden resumir en los siguientes puntos:

- *Fase de preparación*: reunir toda la información y adquirir la base teórica necesaria para poder afrontar la posterior realización del proyecto:
  - Documentación sobre IDEs (Infraestructuras de Datos Espaciales), sobre SIGs (Sistemas de Información Geográfica), proyecciones cartográficas, sistemas de coordenadas y todo el software relacionado con una IDE.
  - Documentación, estudio y elección del software y las tecnologías que se utilizarán para la realización de la aplicación web.
- *Fase de especificación de requisitos*: el principal objetivo de esta etapa consiste en describir el comportamiento del sistema y los componentes que lo forman, de acuerdo a los requisitos solicitados por el cliente final. Para ello se analizarán las principales condiciones que debe reunir el sistema para que realice su función de manera correcta.
  - Descripción detallada del proyecto: reunirá los objetivos, requisitos y restricciones presentes en la aplicación a desarrollar. Exponer de forma que se facilite su posterior validación.
  - Especificación y diseño de la IDE. Estudio de las diferentes soluciones que existen y que mejor se adaptan a la IDE proyectada.
- *Fase de implementación*: el objetivo será implementar el diseño especificado en la fase anterior, centrándose en las siguientes tareas:
  - Instalación en un servidor de todo el software necesario para la IDE. Esto proporcionará la infraestructura necesaria para poder hacer pruebas durante el desarrollo de la aplicación web.
  - Desarrollo de la aplicación web y la interfaz cliente para el cálculo de rutas.
- *Fase de pruebas*: en esta fase se verificará el correcto funcionamiento del sistema, mediante unos casos de prueba que intentarán cubrir el máximo número de posibilidades. Se hacen pruebas de rendimiento del servidor y comparativas, y depuración en los casos necesarios.
- *Fase de documentación*: organizar toda la información recopilada durante el desarrollo del proyecto en la presente memoria.

2.10 Pruebas

Este proyecto incluirá la realización de pruebas para comprobar su correcto funcionamiento. Dichas pruebas consistirán fundamentalmente en comprobaciones y correcciones durante la implementación del sistema, así como validación de los requisitos principales desarrollados (los casos de uso).

Esta parte se describirá detalladamente en el bloque 4 dedicado a las pruebas de la aplicación.

2.11 Seguridad

La seguridad es otro de los componentes esenciales de la aplicación web, que por su propia naturaleza está expuesta a ataques de toda índole a través de las redes. Hay algunos aspectos que dependen del servidor, como los ataques de denegación de servicios (DoS). Para dotar de seguridad a la aplicación se implementarán las siguientes directrices:

- Los datos de acceso a la base de datos residirán en el servidor, y nunca se podrá acceder directamente a la base de datos desde un *script* en el navegador del usuario. Esto se traduce en que el código encargado de negociar con la base de datos será visible sólo por el servidor.
- Las contraseñas de los usuarios serán guardadas en la base de datos de forma segura.
- Los archivos pertenecientes a la IDE y a la aplicación a los que el usuario no necesita acceder directamente, estarán fuera del dominio del servidor.
- Se otorgarán los permisos pertinentes en los directorios de la aplicación dentro del dominio web para que los usuarios no puedan acceder al árbol del directorio y ver qué archivos hay en él.
- Se incluirán mecanismos para ocultar la ruta y el nombre de los archivos implicados en la aplicación web, sin afectar al funcionamiento ni al rendimiento de la misma.
- Se implementará un control de usuarios, dado que habrá tres perfiles de usuario (administrador, usuario autenticado y usuario anónimo), para evitar por ejemplo que un usuario anónimo pueda acceder al panel del administrador.

Con estos puntos se pretenderá dotar de seguridad y proteger a la aplicación web de posibles ataques maliciosos a través de Internet.

CAPÍTULO

3 Antecedentes, estado del arte

En este capítulo se profundizará en los conceptos asociados a la tecnología de los Sistemas de Información Geográfica (SIG) y las Infraestructuras de Datos Espaciales (IDE) definidos en el capítulo de introducción, así como en toda la terminología relacionada con la cartografía, para poner en contexto el proyecto que se pretender realizar.

3.1 Cartografía

La Cartografía se puede definir como el arte y la técnica que, con la ayuda de las ciencias geográficas tiene por objeto el levantamiento, la redacción y la publicación de un mapa. Es decir, es la ciencia encargada de la elaboración y estudio de mapas.

Con el avance tecnológico del último siglo, concretamente en el área de la Informática, la Cartografía ha vivido una revolución a todos los niveles. La digitalización de mapas, visualización y análisis espacial por medio de ordenadores facilitó las actividades comunes de la Cartografía y amplió la capacidad de generación de información cartográfica.

El objetivo final de la Cartografía consiste en representar en un plano una zona más o menos extensa, o la totalidad, de la superficie terrestre. Dado que la superficie de la Tierra es casi esférica, en concreto un geoide (está achatada por los polos, debido a la fuerzas de la gravedad y centrífuga), se hace imprescindible aplicar alguna transformación para lograr la representación en el plano.

En este apartado se van a definir una serie de conceptos de Cartografía que están directamente relacionados con este proyecto [10].

3.1.1 Proyecciones cartográficas

La Cartografía estudia los sistemas de proyección más convenientes para definir de forma biunívoca una relación matemática entre los puntos de la superficie terrestre (considerada casi esférica, un geoide) y sus transformados en el plano. Son las llamadas proyecciones cartográficas. Por tanto, la proyección cartográfica (o proyección geográfica) es un sistema de representación gráfica que establece una relación ordenada entre los puntos de la superficie curva de la Tierra y los de una superficie plana (representada en un mapa).

El proceso seguido hasta llegar a la representación plana de una porción de la superficie de la Tierra comienza con la toma de medidas en la superficie terrestre. A continuación se realizan una serie de correcciones para referir esas medidas al geoide que representa la Tierra, y sobre esos datos se vuelve a hacer correcciones para modelar los datos respecto a un elipsoide de referencia previamente adoptado (una aproximación del geoide terrestre más fácil de manejar y de realizar cálculos sobre él). Tras este proceso, se aplican las relaciones matemáticas definidas por el sistema de proyección elegido a los datos, y el resultado se proyecta en un plano. Todas las proyecciones cartográficas siempre producen alguna deformación en la superficie representada (es lo que se conoce como anamorfosis).

Las proyecciones cartográficas se pueden clasificar en:

- Proyecciones cilíndricas: se proyecta la superficie de la Tierra sobre un cilindro. Hay varios tipos de proyecciones cilíndricas, según la colocación del cilíndro tangente al globo terráqueo, como la cilíndrica regular (tangente en el Ecuador), o cilíndrica transversa (tangente al meridiano central). Dentro de esta última se encuentra la famosa proyección transversa de Mercator (1772), que tras sufrir algunos ajustes por parte de Gauss (1822) y Kruger (1912), ha dado origen al actual sistema de proyección UTM (Universal Transversal de Mercator).

- Proyecciones cónicas: se proyecta sobre un cono tangente (o secante) a la Tierra alrededor de un paralelo (que suele estar situado en una latitud media). No son tan utilizadas como las cilíndricas, dado que la zona de precisión suele ser muy restringida, y las deformaciones fuera de ella son muy notables. Algunos ejemplos de proyecciones cónicas son la cónica simple, la conforme de Lambert, o la equiárea de Albers.

- Proyecciones acimutales (o planares): se sitúa un plano tangente a la Tierra, y se proyecta respecto a un punto seleccionado que hace de fuente de luz que generará la proyección, ya sea en el interior de el Globo (gnomónica) o en el exterior (ortográfica).

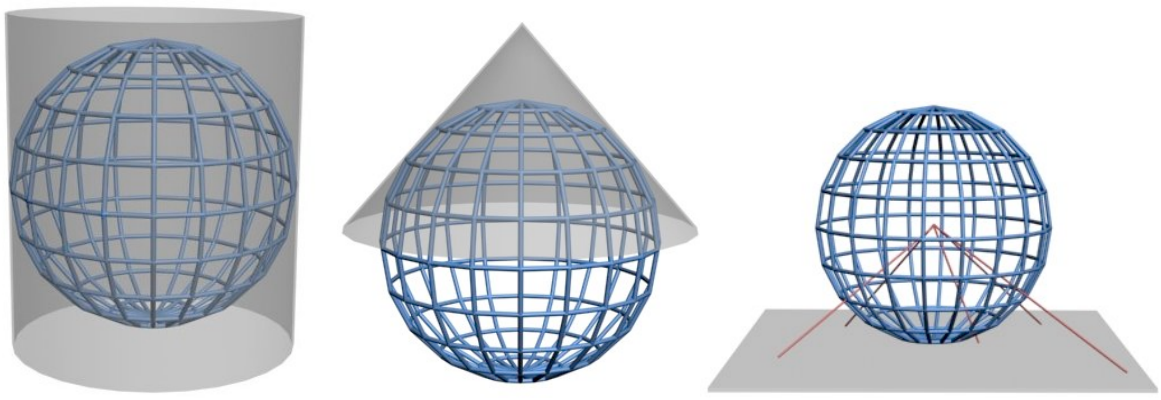

Figura 3.1. Proyección cilíndrica, cónica, y acimutal gnomónica.

Para un sistema de proyección como UTM, se debe partir y acordar tres premisas:

• Elección de un elipsoide de referencia: es el patrón matemático que interviene en la transformación de los datos en el plano. Algunos ejemplos son el elipsoide de Hayford, o el WGS84.

• Elección de unos puntos de referencia del elipsoide (datum): un conjunto de parámetros usados para determinar la posición del elipsoide de referencia con respecto a la superficie real de la Tierra. Los datum clásicos están definidos por 8 parámetros: 6 que definen la posición en el espacio de un elipsoide de referencia y 2 para la forma o tamaño del mismo, fueron concebidos para ajustarse a una porción determinada de la Tierra.

• Elección de un sistema de representación plano conforme: es decir, que conserva los ángulos. Para evitar grandes deformaciones se divide la superficie terrestre en 60 husos (secciones).

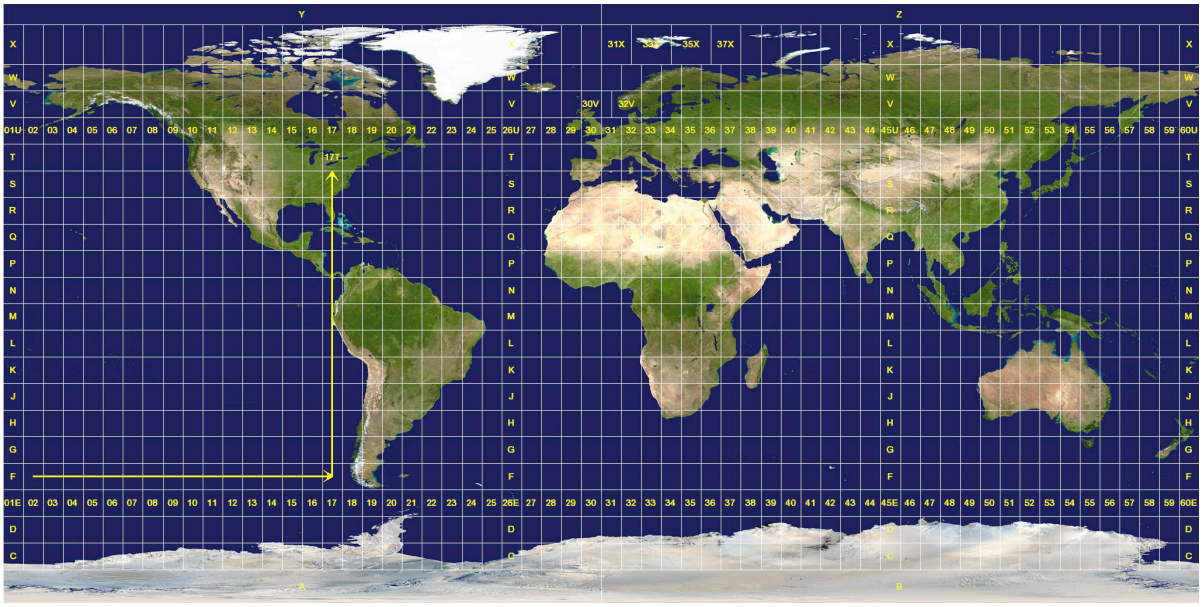

Figura 3.2. Zonas (husos) UTM.

3.1.2 Sistemas de coordenadas

Un sistema de coordenadas es un conjunto de valores que definen la posición de cualquier punto de un espacio vectorial. Existen varios sistemas coordenadas, como las cartesianas, las polares, o las esféricas. En cartografía se usan las coordenadas geográficas, que son un tipo de sistema de coordenadas esféricas que se usa para definir puntos sobre una parte o la totalidad de la superficie terrestre.

El sistema más conocido es el que emplea la longitud y la latitud, cuyos valores se expresan en grados. Por otra parte, también se pueden definir dichas coordenadas mediante proyecciones cartográficas: por ejemplo, el sistema de coordenadas Universal Transversal de Mercator (UTM), una de las más utilizadas. Estas coordenadas cartesianas (x,y), expresadas en metros, hacen referencia al plano transformado, resultado de la proyección del elipsoide de referencia (cuyas coordenadas son geográficas, longitud y latitud).

3.1.3 Escalas

Los mapas, para ser manejables por seres humanos, son más pequeños que las áreas que representan, y por tanto debe haber una forma de señalar la razón entre magnitudes (la representada y la real). Esta razón se llama escala numérica del mapa, que se indica en el mapa mediante una relación numérica. Por ejemplo, una escala 1:5000 indica que 1 cm del mapa equivale a 50 m en la realidad.

La escala numérica será completamente válida sólo en ciertos puntos o a lo largo de determinadas líneas (llamadas automecoicas), en el resto será un valor aproximado. El factor de escala se define como el cociente entre la escala numérica y el valor real de la escala.

3.1.4 European Petroleum Survey Group (EPSG)

En el ámbito de los Sistemas de Información Geográfica existe desde hace tiempo y de forma normalizada (en concreto con la norma ISO 1911) una base de datos que ha codificado en una serie parámetros geodésicos, que definen un sistema de referencia concreto (definido por un código numérico) para su uso informático, datos de sistemas de proyección, sistema de coordenadas, elipsoides, datums y otros datos relacionados con la Cartografía.

De esta forma, queda expresado en un sólo número, de manera unívoca y universal, toda la información relativa al sistema de referencia en el que esté representado un conjunto de datos geoespaciales. Por ejemplo, el código EPSG:25830 hace referencia al sistema con el datum ETRS89, sistema de coordenadas UTM y huso 30 Norte.

El European Petroleum Survey Group (EPSG) fue el impulsor de este modelo, y desde 2005 la OGP (International Association of Oil and Gas Producers) se encarga de mantener actualizada la base de datos, en colaboración con el consorcio OGC [11].

3.2 Sistemas de Información Geográfica (SIG)

Un Sistema de Información Geográfica (SIG) se puede definir como el conjunto integrado de hardware, software y datos geográficos diseñado con objeto de visualizar, almacenar, editar y analizar información geográfica espacial referenciada con la finalidad de afrontar y resolver problemas de gestión, descripción, ordenación y planificación del territorio. En un sentido más genérico, los SIG son herramientas que permiten a los usuarios realizar consultas geográficas de forma interactiva, analizar la información espacial, manipular datos y mapas y presentar los resultados de todas estas operaciones.

Los Sistemas de Información Geográfica se han convertido de una herramienta muy difundida, debido a la cantidad y variedad de actividades en las que pueden tomar parte y ser de utilidad, que se pueden clasificar en los siguientes grupos [2, 3]:

- Gestión y descripción del territorio: trata de referenciar geográficamente la ubicación física de los objetos de estudio, útil por ejemplo para el mantenimiento y control de grandes infraestructuras (red hidrográfica, red telefónica, etc.), la gestión de datos catastrales o la gestión municipal.

- Ordenación y planificación del territorio: trata de estudiar e inferir estrategias que lleguen a describir dónde hay que ubicar los objetos de estudio de acuerdo a unos objetivos. Por ejemplo, la planificación urbana y ambiental, o el análisis y puesta en marcha de políticas relacionadas con el transporte (flujo de tráfico, cálculo de rutas óptimas).

Los datos geoespaciales que puede manejar un SIG se dividen en dos tipos:

- Datos raster: son imágenes digitales, divididas normalmente en celdas regulares organizadas, y se suelen utilizar para representar fotografías aéreas o gráficos representativos del terreno (mapas topográficos, por ejemplo).
- Datos vectoriales: los datos están representados como vectores o puntos con un determinado conjunto de valores, de forma que sobre ellos se podrán realizar operaciones matemáticas complejas.

3.3 Infraestructuras de Datos Espaciales (IDE)

En el capítulo de introducción se definió qué era una IDE y se comentó un punto importante: la interoperabildiad con otras IDEs.

Los objetivos que se buscan con la implantación de las IDEs son principalmente facilitar el acceso a todos los niveles (organizaciones estatales, empresas, particulares) y la integración de la información espacial, que permitirá generalizar el uso de la información geográfica y planificar de forma óptima determinadas actividades que involucren características geográficas, o de localización.

Los avances tecnológicos y la cotidianidad del componente geoespacial en multitud de actividades, han multiplicado el volumen de información geográfica. Las IDEs pretenden organizar, catalogar y poner al alcance de todo el mundo esta información, y dotarla de la máxima utilidad.

En España existe una organización estatal (el Consejo Superior Geográfico) que está llevando a cabo un proyecto (llamado IDEE, Infraestructura de Datos Espaciales de España) coordinado con el resto de países europeos, para integrar ordenadamente y siguiendo unas normas estándar toda la información espacial recopilada del territorio nacional, integrando nodos IDE locales y regionales para construir una gran red IDE nacional [4].

3.4 Web Mapping

El auge de Internet ha alcanzado al ámbito de los Sistemas de Información Geográfica, que se han adaptado para ofrecer aplicaciones con objeto de visualizar y editar información geográfica en entornos web, como por ejemplo Google Maps, OpenStreetMap o Bing Maps. Estas aplicaciones proveen servicios en forma de mapas interactivos, con algunas funciones de enrutamiento.

CAPÍTULO

4 Objetivos

Tras haber descrito el proyecto y todo su entorno en el capítulo anterior, a continuación se exponen los objetivos del mismo.

4.1 Objetivo principal

El objetivo principal del proyecto es proporcionar al usuario, mediante el uso de una aplicación web, la ruta más afín a sus preferencias dentro de las posibilidades contempladas en la Infraestructura de Datos Espaciales (IDE) que se utilizará.

Más concretamente, la herramienta deberá recoger diferentes características (capacidades físicas, intereses, etc.) del usuario y emplearlas para ofrecer las rutas que le sean más apropiadas. Para ello utilizará diversos métodos de búsqueda (camino más corto, optimización multicriterio, etc) que ya están disponibles en la IDE.

4.2 Objetivos específicos

Los objetivos específicos que se pretenden alcanzar con el desarrollo de dicho proyecto:

- Diseñar e instalar una Infraestructura de Datos Espaciales (IDE) que gestione datos de rutas o recorridos para senderismo a partir de la cartografía de la Junta de Andalucía.

- Especificación, diseño y desarrollo de una aplicación web que presente los datos más apropiados para un usuario según su aptitud. Para ello:

- El sistema recogerá los datos necesarios del usuario y los integrará con parámetros calculados y otros posibles criterios específicos

- Consultará en la IDE cuáles son las rutas más apropiadas. Ese cálculo se realizará mediante métodos de búsqueda ya disponibles en la IDE.

- Devolverá y mostrará dichas rutas de forma que sean útiles al usuario.

• Se gestionarán los usuarios que hagan uso del sistema para poder realizar tareas más avanzadas que complementen la búsqueda de rutas de forma anónima (controlar rutas ya realizadas, recuperar información sobre capacidades físicas del usuario, etc.). Se permitirá el acceso a usuarios no registrados, pero al no haber perfil no se guardarán las rutas consultadas.

CAPÍTULO

5 Restricciones

En este capítulo se pondrán de relieve las restricciones o limitaciones que afectan al desarrollo del proyecto, y que consecuentemente influirán en la toma de decisiones. Los factores restrictivos se pueden clasificar en dos grupos:

- *Factores dato*: son aquellos que sobrevienen de forma ajena al desarrollador de la aplicación software, y que deben cumplirse sin posibilidad de ser modificados.

- *Factores estratégicos*: son aquellas decisiones iniciales tenidas en cuenta únicamente por parte del desarrollador, y que por tanto no vienen impuestas externamente.

5.1 Factores dato

En el desarrollo del presente proyecto se identifican los siguientes factores dato:

- Debido a que la aplicación se encuentra relacionada directamente con otro proyecto, deberá adaptarse para dar cabida a ese otro proyecto (algoritmo de búsqueda multicriterio).

- Los datos proporcionados se limitarán a la zona del Parque Natural de la Sierra de las Nieves.

- Los recursos software usados durante el desarrollo del proyecto deberán ser libres y gratuitos, dada la libertad de uso y redistribución, por el soporte y compatibilidad a largo plazo, o por el uso de formatos estándar que proporciona el software libre, y por el que muchas administraciones están apostando. En caso de necesitar el uso de software propietario, se intentará utilizar los que disponga el autor del proyecto o los que suministre el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga.

• Los recursos hardware se limitarán a los disponibles por parte del autor de este proyecto, y los que les sean suministrados por el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga.

5.2 Factores estratégicos

En cuanto a los factores estratégicos y las razones por las que se opta por ellos, vienen reflejados en los siguientes puntos:

• Entorno de desarrollo con software libre, con las ventajas comentadas anteriormente, y que permite disponer de un soporte abierto, con ejemplos código de otros programas (lo que puede facilitar el desarrollo), y una corrección más rápida y eficiente de fallos.

• La aplicación será diseñada de forma modular y usando un modelo basado en capas, buscando que sea fácil de mantener posteriormente y se le puedan aplicar futuras mejoras (escalabilidad).

• Debido a la naturaleza de la aplicación, y su ejecución en el mayor número posible de navegadores, deberá ser *cross-browser*, y seguir todos los estándares fijados por la W3C y la OGC. Así mismo, las pruebas se realizarán sobre varios navegadores web.

• El proceso de desarrollo se hará siguiendo los pasos establecidos en la Ingeniera de Software, y se crearán los diagramas requeridos en las fases de especificación, diseño e implementación.

CAPÍTULO

6 Recursos

Para la elaboración del presente proyecto se tendrá a disposición del desarrollador una serie de recursos, que pueden ser de tres tipos:

- *Recursos humanos*: referidos a toda aquella persona que intervenga en el desarrollo del proyecto.
- *Recursos materiales*: el soporte hardware y software disponibles para la elaboración del proyecto.

6.1 Recursos humanos

El conjunto de personas involucradas en el proyecto son las siguientes:

- Directores del proyecto: debido a la naturaleza multidisciplinar del proyecto, se compone de tres directores con las siguientes funciones:

  ‣ Sebastián Castillo, personal de la Universidad de Málaga, se encargará de guiar al alumno durante el desarrollo del proyecto, supervisando las tareas de desarrollo para comprobar que los resultados que vaya obteniéndose se adecuen a los requisitos especificados.

  ‣ Prof. Dr. José del Campo, del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, se encargará de coordinar todas las actividades del proyecto, así como de gestionar todos los aspectos administrativos y burocráticos necesarios para disponer de un servidor de su departamento donde se pondrá en marcha la IDE proyectada.

  ‣ Prof. Dr. Jesús Vías, del Departamento de Geografía de la Universidad de Málaga, será el encargado de proponer los objetivos que este proyecto desea alcanzar, así como de facilitar los datos geoespaciales para el proyecto.

• Autor del proyecto: Jose Rolland López de Coca, alumno de la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Málaga, será el encargado de planificar el proyecto, desarrollar el sistema software, realizar pruebas al mismo, recopilar toda la información y generar la documentación de todo el proceso.

6.2 Recursos materiales

6.2.1 Recursos hardware

Se dispone de un equipo servidor de los laboratorios del Departamento de Lenguajes y Ciencias de la Computación de la E.T.S. de Informática para montar toda la Infraestructura de Datos Espaciales y la aplicación web, con objeto de realizar pruebas en un entorno real.

Además el alumno contará con un ordenador portátil equipado con un procesador Intel Core 2 Duo a 2 GHz, 2 GB de memoria principal, disco duro SATA de 500 GB, y tarjeta gráfica de 256 MB de VRAM, para llevar a cabo todas las fases del proyecto.

6.2.2 Recursos software

En cuanto a los recursos software empleados para el proyecto cabe destacar lo siguiente:

• Se hará uso de un entorno de desarrollo para la creación y edición del código fuente de la aplicación, y se usarán las herramientas necesarias para llevarlo a cabo.

• Se usará un procesador de textos para la edición de esta memoria.

• Se utilizará alguna herramienta para la creación de diagramas relacionados con el desarrollo del proyecto que aparecerán en esta memoria.

BLOQUE 2: ESPECIFICACIÓN Y ANÁLISIS DEL SISTEMA

CAPÍTULO

7 Análisis de la especificación de requisitos

Para abordar el estudio de la especificación de requisitos, conviene tener presente el siguiente gráfico, que pretende mostrar un esquema aproximado del sistema que se va a desarrollar:

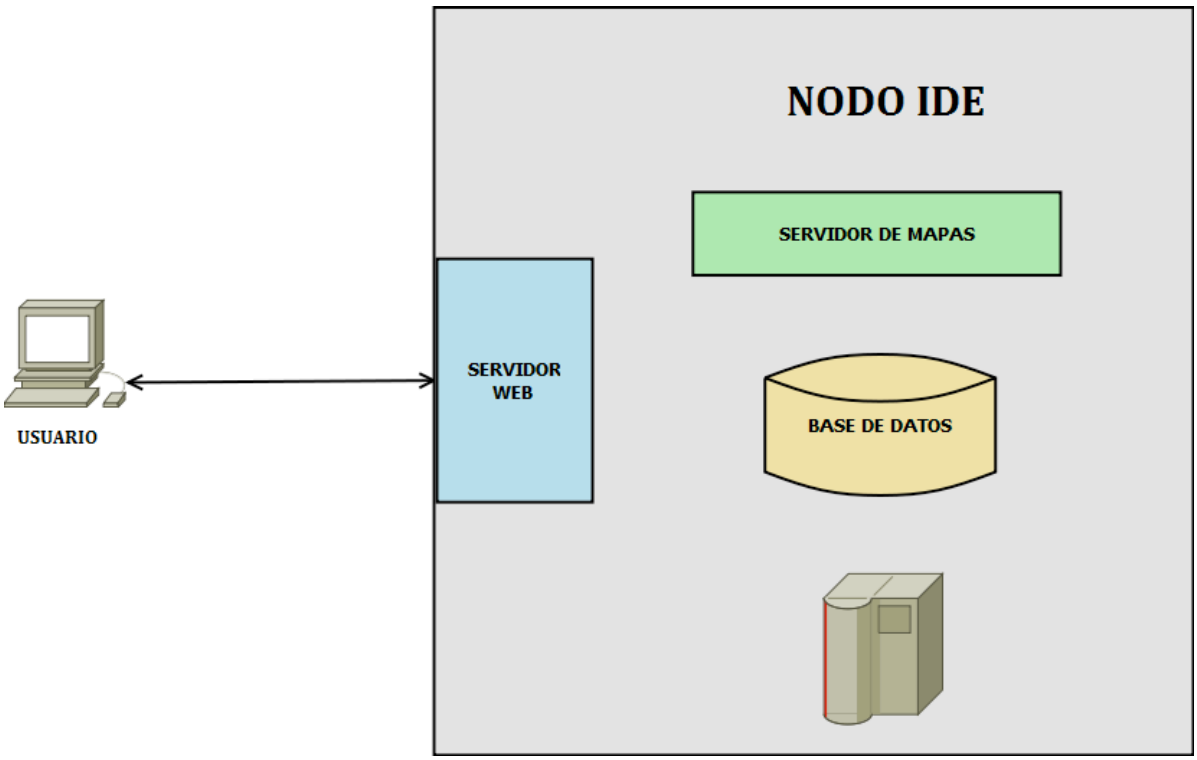

Figura 7.1 Primer esquema del sistema.

Los objetivos ya comentados son crear un nodo IDE y una aplicación web de cálculo de rutas que se conectará a ese nodo cada vez que manipulen datos geoespaciales.

Las actividades relativas a la especificación de requisitos suelen pasar por cinco fases:

- Obtención de requisitos: charlas o entrevistas con el cliente, para conocer sus peticiones.
- Análisis: transformar los deseos del cliente en condiciones apropiadas para ser tratados por el diseño.
- Documentación: los requisitos deben ser documentados.
- Verificación: comprobar el correcto funcionamiento de un requisito en la aplicación.
- Validación: comprobar que los requisitos implementados se corresponden con lo que inicialmente se pretendía.

La especificación de requisitos software aspira a describir el comportamiento del sistema que se va a desarrollar, y vamos diferenciar entre los siguientes tipos de requisitos:

- *Requisitos funcionales*: definen el comportamiento interno del software, manipulación de datos y otras funcionalidades que describen las interacciones que tendrán los usuarios con el software (los casos de uso).
- *Requisitos no funcionales*: son los que juzgan e imponen restricciones al diseño o implementación del sistema (como por ejemplo los estándares de calidad, estabilidad, eficiencia, portabilidad o coste).
- *Usabilidad*: es un aspecto que ha ido cogiendo fuerza en los últimos tiempos, y se ha extendido su definición más allá de relacionarlo con la interfaz gráfica. Tradicionalmente la usabilidad ha sido vista como un requisito no funcional, pero recientemente se ha considerado el impacto que tiene en la funcionalidad del sistema, así que dado ese carácter dual será tratada por separado en un nuevo apartado.

Además, para facilitar la tarea se van a estudiar los casos de uso del comportamiento del sistema, tal y como se expone a continuación.

7.1 Tipos de requisitos

7.1.1 Requisitos funcionales

Tal y como se acaba de comentar, los requisitos funcionales describen cómo debe actuar el sistema. Los diagramas de casos de uso del Lenguaje de Modelado Unificado (UML) son precisamente diagramas de comportamiento del sistema al afrontar un requisito, sin especificar cómo se implementará (es decir, explica *qué* hace, no *cómo* lo hace), y nos ayudarán en la tarea de describir los requisitos funcionales [12].

7.1.2 Requisitos no funcionales

Los requisitos no funcionales añaden ciertas propiedades adicionales a la solución (a veces se dice que estos requisitos provienen del mundo real, fuera del mundo virtual que genera el ordenador), independientes del

correcto funcionamiento de la aplicación. No describen funciones específicas ni información a manipular por parte del sistema, sino características que analicen y juzguen cómo opera el sistema. A continuación se describen los principales requisitos no funcionales de la aplicación:

- Eficiencia y rapidez: el sistema debe ser eficiente, consumir los mínimos recursos posibles, y responder a las peticiones del usuario de la forma más rápida posible, para una mayor agilidad en la comunicación entre usuario y sistema informático.

- Fiabilidad y disponibilidad: el sistema también debe ser robusto e interactuar con el usuario de la forma más rápida pero estable posible. Además, siempre deberá estar disponible de cara al usuario.

- Tolerancia a fallos: el sistema debe responder frente a posibles errores por parte suya o del usuario, y recuperarse continuando el flujo de ejecución.

- Estándares: el sistema cumplirá con todos los estándares relativos a su ámbito, para asegurar la calidad.

- Portabilidad: el sistema deberá poder adaptarse a nuevos entornos y circunstancias. Por ejemplo, el uso de direcciones URL relativas dentro del código facilita pasar de un dominio web a otro sin tener que modificar el código fuente.

- Seguridad: el sistema deberá ser mínimamente seguro, y salvaguardar la información sensible.

- Mantenimiento: el sistema deberá ser fácil de mantener, para poder ser actualizado en el futuro.

- Coste: no se podrá sobrepasar el presupuesto destinado al proyecto.

7.1.3 Usabilidad

La usabilidad se define formalmente como la eficacia, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios concretos utilizando un determinado sistema (ISO 9241, 1998). En un sistema interactivo la usabilidad se presenta como la característica esencial del interfaz entre usuario y máquina para que aquél lo use de la forma más rápida y fácil, y hay que tenerla en cuenta desde el principio, en las primeras fases de desarrollo de una aplicación software. Tal importancia ha adquirido la usabilidad que se habla de una Ingeniería de Usabilidad, una práctica que agrupa todas las técnicas necesarias referentes a leyes de usabilidad, heurísticas y desarrollo de pruebas de usabilidad.

La definición anterior puede resultar muy teórica, así que algunos autores han añadido a esa definición algunos indicadores de usabilidad más concretos que ayuden a evaluar el interfaz, destacando los siguientes:

- Eficiencia: es deseable que el sistema sea lo más eficiente posible, y así pueda repercutir en una mayor productividad por parte del usuario.

- Facilidad para memorizar el interfaz (*memorability*): un interfaz fácil de manejar y sencillo de recordar por parte del usuario mejora la usabilidad del sistema, permitiendo al usuario utilizarlo con mayor agilidad, incluso después de un tiempo sin haberlo usado.

- Facilidad para aprender a usar el sistema (*learnability*): lograr que el usuario aprenda a usar el sistema de forma fácil y rápida, y pueda comenzar a usarlo lo antes posible, aumenta la usabilidad.

- Satisfacción: un sistema usable debe proporcionar a los usuarios un grado de satisfacción alto.

- Ausencia de errores: evitar algunos errores del usuario es tarea imposible (como por ejemplo introducir mal la contraseña), pero la usabilidad del sistema aumentará si se pueden evitar incluir acciones en la interfaz que puedan provocar errores.

Para un sitio web es importante tener en cuenta estos otros puntos:

- Consistencia: es deseable que el sistema muestre un diseño consistente (que todas las áreas tengan el mismo diseño), para facilitar al usuario el uso del sitio web.

- Reversibilidad: un sitio web debe permitir deshacer las acciones realizadas, siempre que sea posible.

Aunque otros autores proponen agrupar los conceptos que rodean a la usabilidad en tres tipos: facilidad para aprender (*learnability*, ya comentado), flexibilidad (un sistema que deja varias posibilidades al usuario de poder manejarlo, será más flexible) y robustez.

Para medir el grado de usabilidad de una aplicación hay que diferenciar entre medidas cualitativas y cuantitativas. A lo largo de los años, las cuantitivas han resultado ser las más útiles para definir diseños de interfaces usables, mientras que las cualitativas, por su propia naturaleza (percepciones, preferencias, opiniones de los usuarios), son más subjetivas y difíciles de estructurar. En el caso de una aplicación web, las medidas cuantitativas suelen ser de dos tipos:

- Relativas a la composición de la página: tamaño de la página, número de palabras, número de enlaces, proporción entre imágenes y texto, scroll, resoluciones, etc.

- Relativas al formato de la página: paleta de colores, tipos de letra, número de párrafos, etc.

Con estas leyes de usabilidad presentes en la fase de especificación, se podrán afrontar futuras pruebas de usabilidad durante las fases de diseño e implementación de la aplicación. Esta evaluación de la usabilidad se puede hacer usando diferentes métodos, dependiendo del tiempo y presupuesto disponibles:

- Métodos de inspección: normalmente guiados por un experto en usabilidad, examinan todos los aspectos relacionados con la usabilidad y emiten un informe con los errores encontrados. Algunos de estos métodos son la evaluación heurística, inspección formal, o inspección de consistencia y estándares.

- Métodos de investigación: recogen opiniones de usuarios relativos a la interfaz, como su claridad o su utilidad. Dependiendo de la manera en la que se produzca la aproximación al usuario para conocer su opinión, se pueden agrupar en: aproximaciones de campo, en grupo o individuales.

- Pruebas de usabilidad: conjunto de experimentos que se realizan sobre usuarios finales para obtener información sobre la interfaz y su usabilidad.

7.2 Casos de uso

Cabe diferenciar entre casos de uso y diagramas de casos de uso: los diagramas son un resumen gráfico general de los casos de uso o un subconjunto de ellos (algo especialmente útil cuando hay demasiados casos de uso y resulta difícil situarlos y relacionarlos), mientras que los casos de uso deben ser descritos en detalle uno a uno mediante un documento o tabla que explique la forma de interactuar entre el sistema y el usuario. Como el lenguaje de modelado UML no define ningún modelo de tabla o documento a este respecto, se procede a definir el modelo de tabla que describa cada caso de uso:

Caso de uso <nº>	Nombre del caso de uso.
Actores	Usuarios que intervienen en el caso de uso.
Objetivo	Breve descripción del objetivo que se desea alcanzar.
Precondiciones	Condiciones previas que debe cumplir para el flujo pueda ser llevado a cabo.
Flujo general	Secuencia de pasos que debe realizar.
Postcondiciones	Condiciones que debe cumplir si el flujo se ha ejecutado correctamente.

Con los casos de uso que vienen a continuación se pretende comenzar a entender cómo funciona el sistema, sin llegar a describirlo absolutamente, ya que nos estamos ciñendo a los aspectos funcionales del mismo.

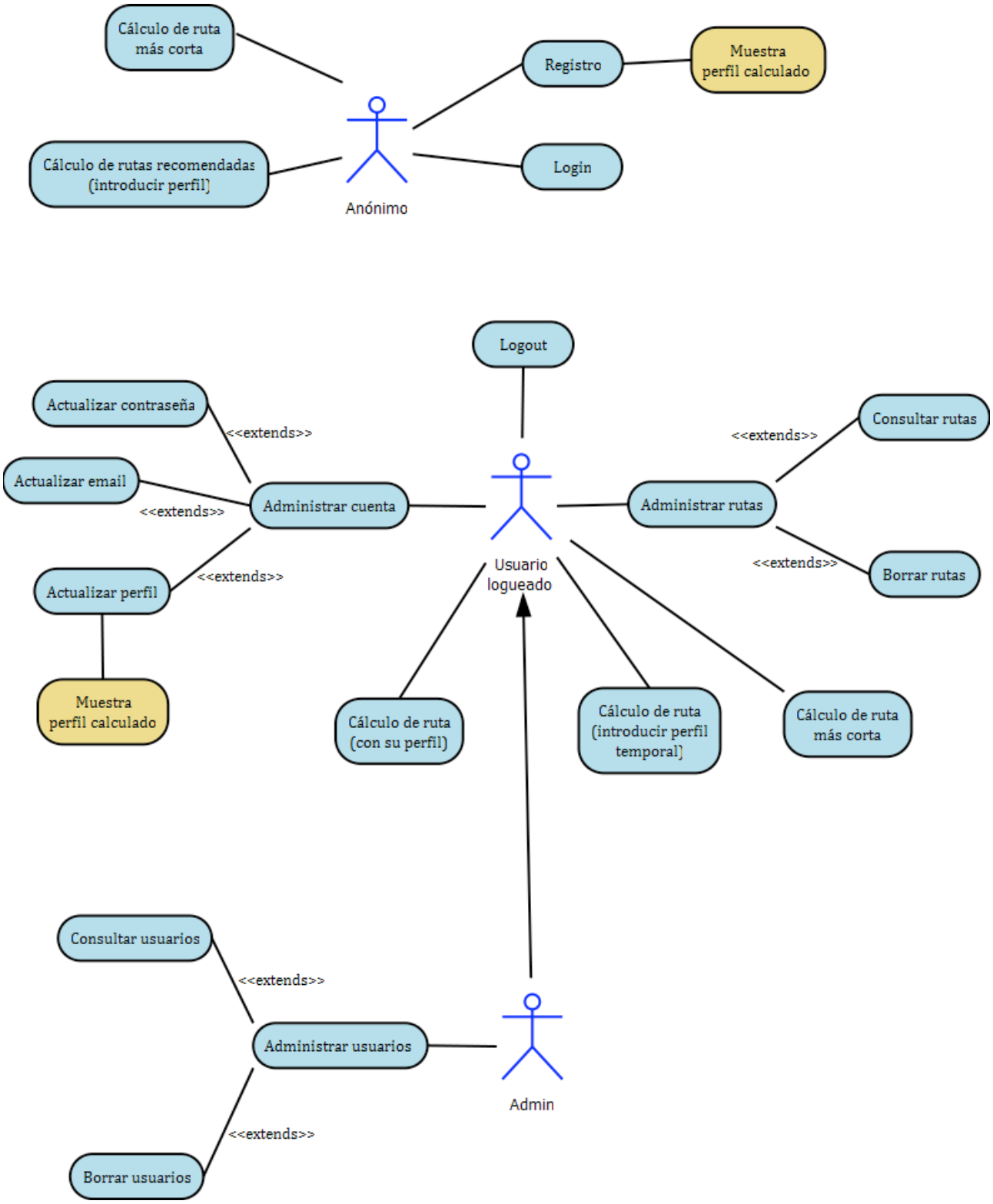

Figura 7.2. Esquema general de los casos de uso.

Caso de uso 1	Registrarse en el sistema.
Actores	Usuario anónimo.
Objetivo	Permite que un usuario se registre en el sistema, y pueda acceder a las ventajas de pertenecer a la comunidad.
Precondiciones	El usuario debe ser anónimo, es decir, que no haya iniciado sesión.
Flujo general	El usuario accederá a la página de registro e introducirá los datos requeridos. El sistema registrará al usuario en la base de datos, e iniciará la sesión del usuario automáticamente.
Postcondiciones	El usuario ha sido registrado en el sistema y con sesión iniciada, y aparecerá un mensaje de confirmación.

Caso de uso 2	Iniciar sesión.
Actores	Usuario anónimo.
Objetivo	Permite a un usuario autenticarse en el sistema.
Precondiciones	El usuario debe estar registrado para poder iniciar sesión.
Flujo general	El usuario introducirá y su nombre y contraseña y procederá a iniciar sesión. El sistema comprobará que los datos son correctos y el usuario está registrado en el sistema: 1-Si son correctos, iniciará la sesión del usuario. 2-Si no, mostrará un mensaje de error.
Postcondiciones	1- El usuario está autenticado. 2- El sistema muestra un mensaje de error.

Caso de uso 3	Editar perfil.
Actores	Usuario autenticado.
Objetivo	Permite recalcular el perfil físico del usuario si éste considera que sus condiciones y aptitudes físicas han cambiado.
Precondiciones	El usuario debe haber iniciado sesión.
Flujo general	El usuario entrará en el panel de control, y a continuación en la página de edición de perfil, e introducirá los datos requeridos. El sistema calculará el nuevo perfil y lo guardará en la base de datos.
Postcondiciones	El nuevo perfil se guardará en la base de datos, y aparecerá un mensaje de confirmación.

Para calcular el perfil del usuario (casos de uso 1 y 3), se tienen en cuenta varios parámetros: si el usuario entrena (realiza ejercicio físico) regularmente o no, su número de pulsaciones por minuto, su edad, su peso, y su altura.

De esos parámetros se obtienen el índice de masa corporal (IMC) y el porcentaje de pulsaciones, de acuerdo las siguientes fórmulas matemáticas:

$$IMC = \frac{peso}{altura^2} \cdot 10000$$

$$\%\,pulsaciones = \frac{núm.\,de\,pulsaciones\,por\,minuto}{(200 - edad)} \cdot 100$$

Las unidades son las siguientes:

- La unidad del peso es el Kilogramo.
- La unidad de la altura es el centímetro.
- La unidad de la edad son los años.

Después de un proceso de extracción de características, se extrajeron las siguientes reglas:

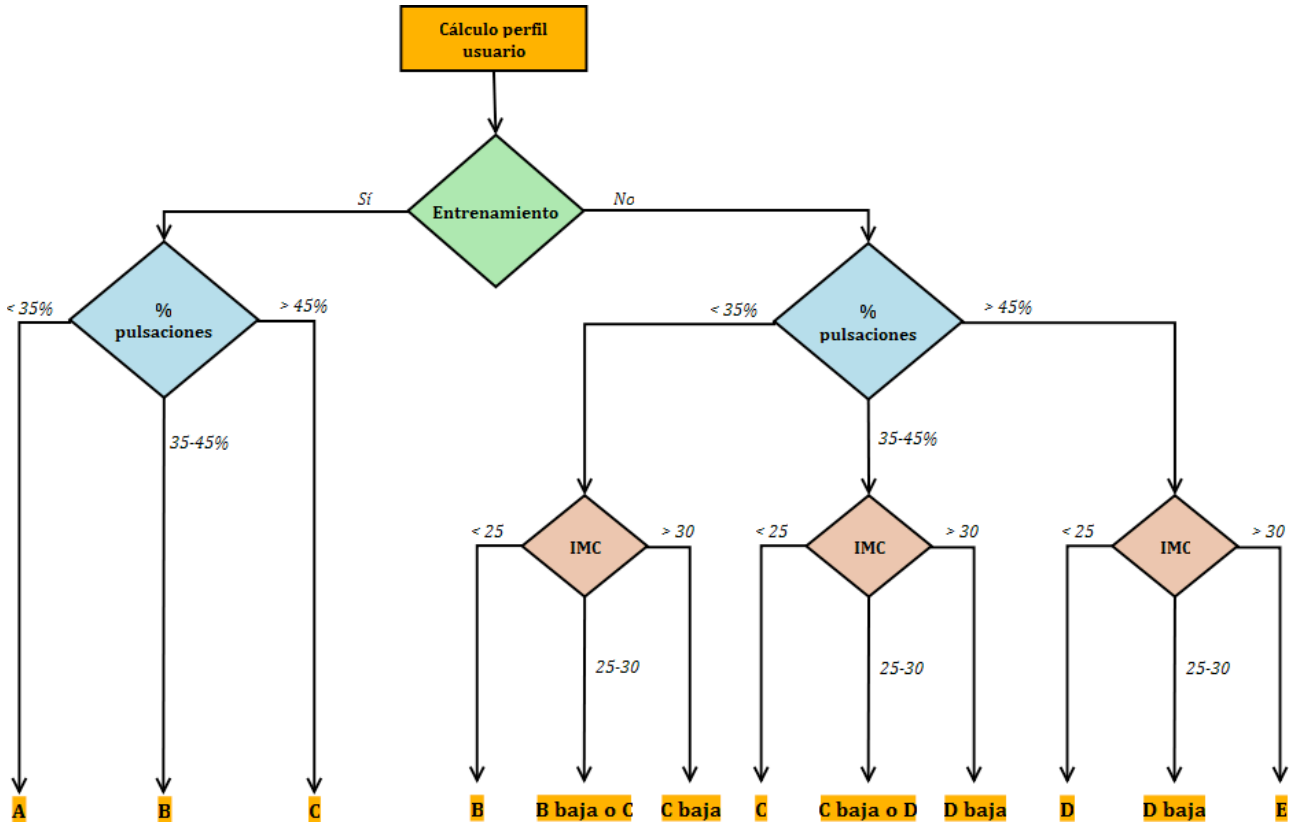

Figura 7.3. Diagrama de cálculo de perfil.

También se puede expresar como una tabla:

Entrenamiento	% pulsaciones	Perfil del usuario		
		IMC		
		< 25	25-30	> 30
Sí	< 35 %	A	A	A
	35 - 45 %	B	B	B
	> 45 %	C	C	C
No	< 35 %	B	B baja o C	C baja
	35 - 45 %	C	C baja o D	D baja
	> 45 %	D	D baja	E

Los distintos perfiles significan lo siguiente (el perfil E es igual que D, pero sólo para motivación de paseo):

- A: rutas de hasta 9,5 horas, con 2700 m. de desnivel, y recorrido máximo de 42 km.
- B: rutas de hasta 9 horas, con 1900 m. de desnivel, y recorrido máximo de 32 km.
- C: rutas de hasta 8,5 horas, con 800 m. de desnivel, y recorrido máximo de 23 km.
- D: rutas de hasta 6 horas, con 500 m. de desnivel, y recorrido máximo de 12 km.

Caso de uso 4	Editar datos personales.
Actores	Usuario autenticado.
Objetivo	Permite editar el correo electrónico y/o la contraseña actuales.
Precondiciones	El usuario debe haber iniciado sesión.
Flujo general	El usuario entrará en el panel de control, y a continuación en la edición de datos personales, e introducirá los datos requeridos. El sistema guardará la nueva información en la base de datos.
Postcondiciones	El nuevo correo electrónico y/o contraseña se guardará en la base de datos, y aparecerá un mensaje de confirmación.

Caso de uso 5	Buscar rutas recomendadas.
Actores	Usuario anónimo, usuario autenticado.
Objetivo	Pregunta al usuario qué ruta desea buscar de acuerdo al perfil, motivaciones y horas disponibles indicados.
Precondiciones	Ninguna.
Flujo general	El usuario accederá desde "Buscador de rutas" a la página de rutas recomendadas. El sistema le preguntará por el perfil, la motivación y opcionalmente las horas disponibles y el fecha prevista para hacer la ruta. El usuario introducirá los datos deseados, el sistema calculará la ruta o las rutas y mostrará los resultados en un mapa y una tabla con las rutas obtenidas.
Postcondiciones	El sistema muestra la página con el mapa y las rutas obtenidas en el cálculo.

El cálculo de rutas recomendadas tendrá en cuenta el perfil del usuario, las motivaciones, y opcionalmente el tiempo disponible.

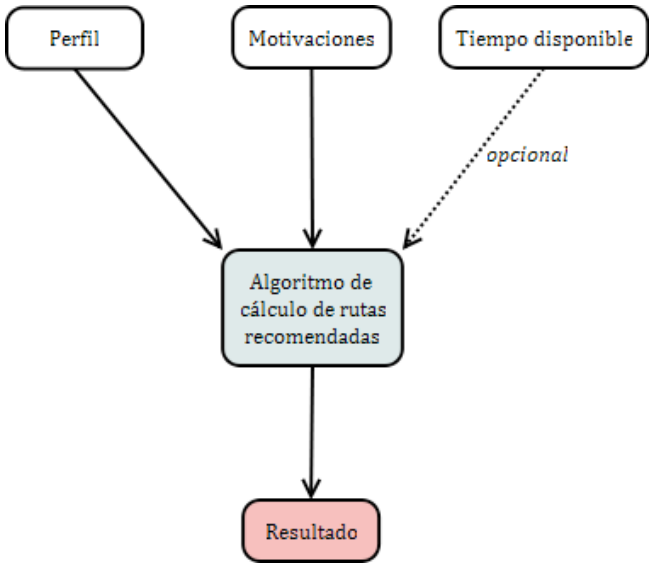

Figura 7.4. Diagrama de las variables que participan en el cálculo de rutas recomendadas.

Existen cuatro posibles motivaciones: deporte, paisaje, ecológico, y paseo. La motivaciones "Deporte" y "Paseo" son excluyentes, es decir, un usuario no podrá buscar una ruta que tenga como motivación ser deportiva y de paseo.

Caso de uso 6	Buscar ruta más corta.
Actores	Usuario anónimo, usuario autenticado.
Objetivo	Ofrecer al usuario la ruta más corta entre dos puntos indicados en el mapa por el usuario.
Precondiciones	Ninguna.
Flujo general	El usuario accederá desde "Buscador de rutas" a la página de ruta más corta. El sistema le mostrará una nueva página con un menú y un mapa donde podrá seleccionar el punto de origen y destino. El sistema calculará la ruta más corta cuando lo indique el usuario y la mostrará en el mapa, suministrando información adicional sobre la misma.
Postcondiciones	El sistema muestra el resultado en el mapa.

Caso de uso 7	Agregar ruta.
Actores	Usuario autenticado.
Objetivo	Permite agregar la ruta seleccionada en una lista personal de rutas del usuario.
Precondiciones	El usuario debe haber iniciado sesión, buscado rutas recomendadas, y encontrarse en la página de resultados.
Flujo general	El usuario pulsará el botón de agregar ruta del sendero que quiere guardar en su lista. El sistema lo almacenará en su lista y mostrará un mensaje de confirmación.
Postcondiciones	La ruta se guardará en la lista de rutas del usuario (dentro de la base de datos), y aparecerá un mensaje de confirmación.

Caso de uso 8	Descargar ruta.
Actores	Usuario autenticado.
Objetivo	Permite descargar la ruta seleccionada en el ordenador del usuario.
Precondiciones	El usuario debe haber iniciado sesión, y o bien encontrarse en la página de gestión de rutas, o bien haber buscado rutas recomendadas, y encontrarse en la página de resultados, o bien encontrarse en la página de ruta más corta.
Flujo general	El usuario pulsará el botón de descargar ruta del sendero que quiere guardar. El sistema le enviará al usuario la ruta.
Postcondiciones	Aparecerá una ventana para descargar el archivo correspondiente a la ruta seleccionada, o se descargará automáticamente (dependiendo de la configuración del navegador web del usuario).

Caso de uso 9	Borrar ruta.
Actores	Usuario autenticado.
Objetivo	Permite borrar una ruta seleccionada de la lista personal de rutas del usuario.
Precondiciones	El usuario debe haber iniciado sesión, y encontrarse en la página de gestión de rutas.
Flujo general	El usuario pulsará el botón de borrar ruta del sendero que quiere borrar de su lista. El sistema mostrará un mensaje de advertencia que el usuario debe confirmar si quiere proceder al borrado. En caso afirmativo el sistema lo borrará de su lista y mostrará un mensaje de confirmación.
Postcondiciones	La ruta será eliminada de la lista de rutas del usuario (dentro de la base de datos), y aparecerá un mensaje de confirmación.

Caso de uso 10	Consultar usuarios del sistema.
Actores	Administrador.
Objetivo	Permite consultar usuarios del sistema.
Precondiciones	El usuario debe haber iniciado sesión como administrador, y encontrarse en la página de gestión del administrador.
Flujo general	El usuario accede a la página de gestión del administrador, donde se mostrará una lista con los usuarios registrados en el sistema, y algunos de sus datos personales.
Postcondiciones	Ninguna.

Caso de uso 11	Borrar usuario.
Actores	Administrador.
Objetivo	Permite consultar, editar o borrar usuarios del sistema.
Precondiciones	El usuario debe haber iniciado sesión como administrador, y encontrarse en la página de gestión de usuarios.
Flujo general	El usuario accede a la página de gestión del administrador, y borrará el usuario seleccionado. El sistema mostrará un mensaje de advertencia que el usuario debe confirmar si quiere proceder al borrado. En caso afirmativo el usuario seleccionado será eliminado del sistema.
Postcondiciones	El usuario será eliminado del sistema, y se mostrará un mensaje de confirmación.

CAPÍTULO

8 Análisis funcional

8.1 Arquitectura lógica del sistema

La arquitectura lógica, también llamada del Software, es el diseño de más alto nivel de la estructura de un sistema, y proporciona el marco de referencia imprescindible para el desarrollo software de un sistema de información. Así, la arquitectura lógica de nuestro sistema queda representada de la siguiente forma:

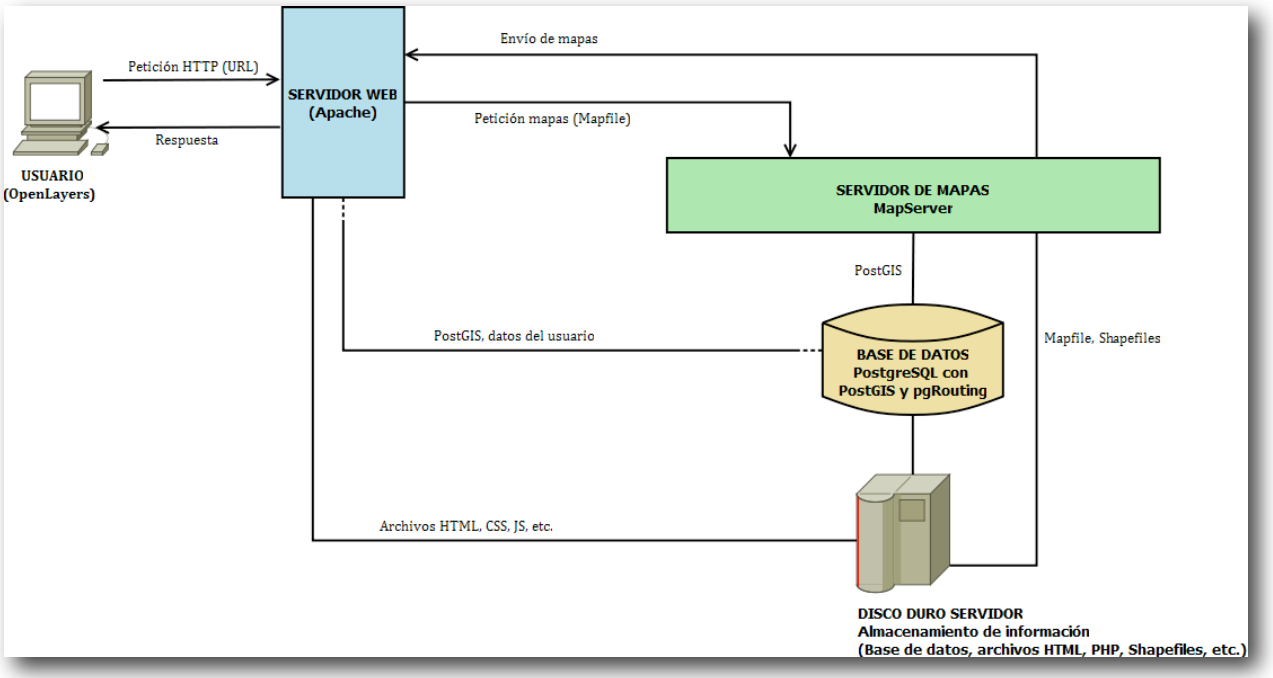

Figura 8.1: Estructura lógica del sistema.

La figura 8.1 representa por tanto un esquema general del sistema que se va a implementar. Tenemos lo siguiente:

- Un servidor web (Apache) [13].
- Un servidor de mapas (UMN MapServer) [14].
- Una base de datos geoespacial (PostgreSQL con PostGIS) [15, 16].
- Almacenamiento de datos en el servidor.
- El usuario que de forma remota (Internet) se conecta al sistema a través de un navegador web.

En definitiva, se trata de una arquitectura basada en un modelo cliente-servidor.

El servidor web es el encargado de recibir peticiones de fuera y responder a ellas, y según la naturaleza de las peticiones necesitará comunicarse con otros módulos del sistema para generar la respuesta adecuada.

El servidor de mapas cumple una función clara: atender las peticiones reclamadas por el servidor web relativas a la visualización web (en forma de imágenes) de mapas.

La base de datos guarda información relativa de los usuarios y también información geoespacial, con la cual podrá analizar y realizar cálculos complejos gracias a la librería pgRouting [17].

Por último, todos los archivos implicados en el sistema estarán almacenados en la memoria del servidor, y podrán ser reclamados tanto por el servidor web (documentos HTML, CSS, JS, etc.) como por el servidor de mapas (archivo de configuración de mapas -Mapfile-, por ejemplo), y por supuesto la base de datos.

8.2 Componentes del sistema

En el presente apartado se detallarán los componentes que intervienen en el sistema anteriormente descrito, y se justificará su inclusión en él.

8.2.1 Servidor Web HTTP: Apache

Un servidor web HTTP es el componente fundamental para dar servicio HTTP (protocolo fundamental usado en cada transacción de la Web, y desarrollado por el W3C) a la aplicación web tanto en una Intranet como en Internet.

La elección del servidor HTTP Apache sobre otras alternativas se fundamenta principalmente sobre estos dos pilares [13]:

- Es de código abierto, gratuito, y multiplataforma (UNIX, Windows, Macintosh).
- Goza de una amplia aceptación en la red, y se estima que es utilizado en casi el 70% de los sitios web en todo el mundo. Esta popularidad también implica una mayor facilidad a la hora de encontrar ayuda y soporte sobre Apache.

8.2.2 Servidor de mapas: UMN MapServer

Un servidor de mapas de Internet (IMS, las siglas en inglés) es un servicio que provee mapas a través de la red, normalmente como imágenes. Una especificación estándar para este tipo de servidores es el OGC Web Map Service (WMS).

UMN MapServer 5.6 es un entorno de desarrollo libre y de código abierto perteneciente al proyecto OSGeo que junto con el servidor web Apache conforma un servidor de mapas, y se muestra como una pieza fundamental en la creación de aplicaciones SIG en Internet con el fin de realizar consultas, visualizar y analizar información geográfica a través de la Red.

Por tanto, este servidor de mapas va a ser la plataforma encargada de la publicación de datos espaciales en forma de mapas interactivos en la web. Mapserver ayudará a servir mapas a la aplicación o a usuarios externos que se conecten al nodo IDE (mediante el protocolo WMS).

UMN MapServer fue desarrollado originalmente en 1994 por el proyecto ForNet de la Universidad de Minnesota (UMN) en colaboración con la NASA y el Minnesota Department of Natural Resources (MNDNR). Más tarde pasó a manos del proyecto TerraSIP, y finalmente ha ido a parar al consorcio OSGeo, donde es mantenido y actualizado periódicamente por un grupo de 20 desarrolladores de todo el mundo.

La siguiente figura es un esquema general del funcionamiento MapServer [14]:

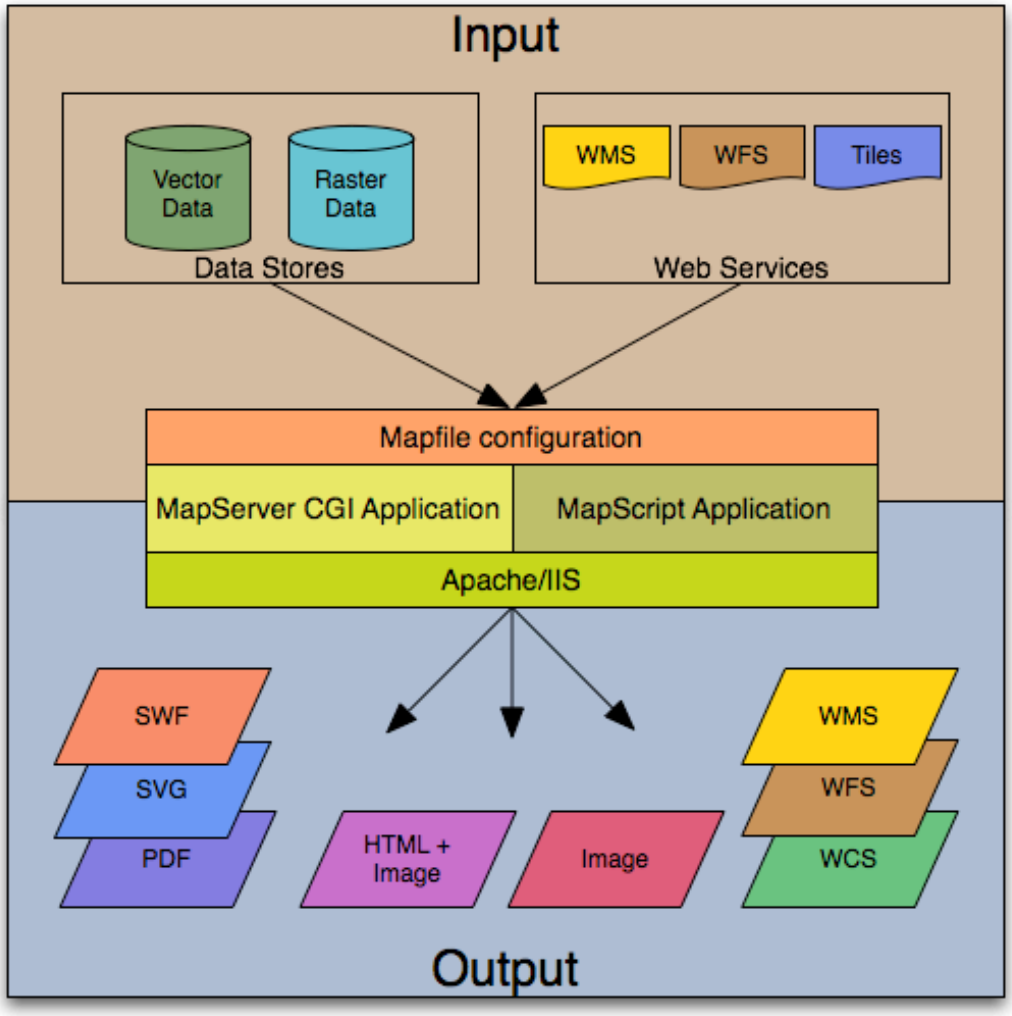

Figura 8.2: Esquema de funcionamiento de MapServer.

La aplicación MapServer funciona básicamente como un programa CGI (Common Gateway Interface, o interfaz de entrada común) que reside en el servidor, y se ejecuta (en el servidor) cuando un cliente (navegador web) así lo solicita.

El funcionamiento de MapServer está configurado en un fichero de texto con extensión ".map" llamado *Map File*. En este archivo, los datos del mapa están organizados en capas, cada una de las cuales puede ser personalizada mediante ciertos parámetros definidos en el lenguaje de configuración propio que usa MapServer (nombre de la capa, escalas, resolución, tipo de datos, proyección), y dentro de ellas se pueden crear clases para definir el aspecto visual. Esta estructura proporciona una personalización de estilos visuales muy flexible.

Además, MapServer facilita una API llamada MapScript que a su vez proporciona un nivel de abstracción mayor hacia otros lenguajes de programación como PHP, Perl, Ruby, Java o C#, de forma que por ejemplo desde una aplicación web escrita en PHP podamos añadir una nueva capa o editar los parámetros de alguna existente, sin tener que acceder directamente al Mapfile.

Las características que apoyan la elección de UMN MapServer y lo distinguen sobre otras opciones, como pueden ser GeoServer o Degree, son las siguientes:

- Salida cartográfica avanzada.
- Alta velocidad de acceso a datos.
- Alta capacidad de personalización.
- Soporte a entornos de desarrollo y *scripting* (PHP, Python, Ruby, Java, Perl, C#).
- Soporte a varios estándares OGC (WMS, WFS, GML, entre otros).
- Variedad de formatos vectoriales soportados: ESRI Shapefiles, PostGIS, MySQL, Oracle Spatial, y otros usando la librería OGR.
- Variedad de formatos raster soportados: PNG, TIF/GeoTIFF, JPEG, EPPL7, y otros usando la librería GDAL.
- Soporte a proyección de mapas (proyección al vuelo de mapas con más de 1000 proyecciones distintas, mediante la librería Proj.4).
- Funciona bajo plataformas Linux, Windows, MacOS X, Solaris, y otros.

8.2.3 Base de datos: PostgreSQL y PostGIS

Existe una gran oferta de base de datos, tanto comerciales como gratuitas y de código abierto. Dado que en este proyecto se va a optar por la segunda vía, quedan dos grandes candidatos: MySQL y PostgreSQL.

MySQL es la base de datos más popular dentro del ámbito del software libre, especialmente en combinación con el servidor web Apache y el lenguaje de programación PHP. En el ámbito geoespacial es mucho menos utilizada ya que no cumple con todos los estándares internacionales de la OGC, ni incorpora toda la funcionalidad y potencia que ofrece PostGIS. Esto último es el principal motivo de haber escogido PostgreSQL con su extensión PostGIS, dada la naturaleza del proyecto.

La base de datos PostgreSQL (versión 8.4), pese a no ser tan popular, está ganando fuerza poco a poco entre la comunidad del software libre y cuenta con un importante equipo de desarrolladores. En algunos estudios ha demostrado ser más robusta que MySQL, mientras que las virtudes de MySQL han residido tradicionalmente en ser más veloz y fácil de usar. No obstante, estos juicios de valor dependen de muchos

factores (como la cantidad de datos y de relaciones, o el entorno tecnológico en el que estén desplegadas las bases de datos), y con cada actualización se han ido corrigiendo esas "debilidades" hasta converger hacia un rendimiento similar entre ambas bases de datos.

El módulo PostGIS (versión 1.5.1), desarrollado principalmente por Refractions Research Inc. bajo la licencia GPL, proporciona a PostgreSQL la capacidad de almacenar información geoespacial (cumpliendo el estándar SFSS, certificado en 2006 por el OGC, lo que garantiza la interoperabilidad con otros sistemas estándar de información geográfica) y de realizar operaciones de análisis geográfico, convirtiéndola en una base de datos espacial para su utilización en Sistemas de Información Geográfica.

8.2.4 La librería pgRouting

Esta librería reside dentro de la base de datos como un conjunto de funciones que pueden ser invocadas en cualquier consulta SQL, y tiene como objetivo proveer a PostgreSQL y PostGIS con funcionalidades avanzadas de enrutamiento. De esta forma, se podrán realizar cálculos de rutas entre dos puntos, usando los datos geoespaciales almacenados en la base de datos.

La librería pgRouting es capaz de transformar un conjunto de datos geoespaciales vectoriales, y construir una topología en forma de grafo de los mismos, con unos nodos que se corresponderán con las interesecciones de líneas unidos por arista, que serán esas líneas y cuyo peso será normalmente la distancia.

PgRouting incorpora tres algoritmos para el cálculo de la ruta más corta (o económica):

- Algoritmo de Dijkstra: diseñado por Edsger Dijkstra en 1959, se trata del algoritmo clásico para este tipo de problemas. Explora todos los caminos más cortos de un grafo desde un nodo origen a un nodo destino, hasta obtener el más corto de todos ellos.

- Algoritmo A*: es una extensión del algoritmo de Dijkstra. Incluye una serie de funciones heurísticas (que estiman el coste del camino más corto) que mejoran la velocidad de cálculo, a costa de perder un poco de precisión, aunque si delimitan bien la zona de cálculo (mediante un cuadro o caja bidimensional) puede obtener resultados similares a Dijkstra. Aplicaciones como Google Maps utilizan este algoritmo.

- El algoritmo Shooting Star: calcula la ruta más corta entre dos aristas, en vez de entre dos nodos, como los anteriores algoritmos. Además, permite añadir restricciones a la búsqueda, como dificultades que puede haber en el camino (tipo de vía, semáforo, obras, etc.).

8.2.5 OpenLayers

Hasta ahora se han visto las herramientas necesarias para publicar un mapa en el nodo IDE, gracias al servidor de mapas y la base de datos. El siguiente paso es poder trabajar con ese mapa, de solicitar información del mismo, visualizarlo, en definitiva de interactuar con él. Existen dos formas de acceder a estos propósitos:

- Mediante una aplicación SIG de escritorio, como gvSIG, Quantum GIS o ArcInfo.

- Mediante alguna herramienta que se pueda incluir en la aplicación web para poder mostrar mapas interactivos en un navegador, que es lo que se conoce como cliente Web-GIS ligero. Entre otros ejemplos, está OpenLayers [18].

OpenLayers es una librería de código abierto escrita en JavaScript para cargar, renderizar y mostrar datos de mapas en navegadores webs modernos, sin depender del lado del servidor (es decir, todo el trabajo de

renderizado lo hace el cliente por medio de su navegador), similar a las APIs de Google Maps o Microsoft Bing, con la diferencia de que OpenLayers es software totalmente libre, y además pertenece al proyecto OSGeo. OpenLayers ha sido desarrollado para poder usar numerosos tipos y formatos de información geográfica, soportando los estándares OGC como WMS o WFS.

Figura 9.3 Aspecto visual de OpenLayers en un navegador web.

Son muchas las ventajas que ofrece este cliente Web-GIS sobre otros de su estilo, como Ka-Map o Mapbender, y que lo han convertido en el más usado y popular de la comunidad web (es entre otros el visor de OpenStreetMap.org, una especie de versión abierta y colaborativa de Google Maps):

- Simplicidad de uso.
- Soporte de teselas (tiles) y cacheado de mapas, que permiten cachear peticiones a servidores de mapas de forma que los clientes reciben teselas para ser visualizadas sin tener que ir directamente al origen de los datos. Esto incrementa notablemente el rendimiento.
- Soporte de estándares OGC.
- Acceso de los mapas de Google Maps, Bing Maps o Yahoo Maps.
- Comunidad de desarrolladores muy activa, con constantes actualizaciones y mejoras.

Por tanto, esta librería se encargará de la tarea fundamental de mostrar mapas y también las rutas ofrecidas al usuario en distintas capas, para que puedan ser manejadas de forma independiente.

8.3 Descripción del comportamiento del sistema

A continuación se explicará el funcionamiento y las operaciones del sistema, y para ello se hará uso principalmente de los diagramas de secuencia. Se obtienen de forma casi automática a partir de los casos de uso expuestos anteriormente, así que cada uno de estos tendrá un diagrama de secuencia asociado.

8.3.1 Diagrama de secuencia 1: registrarse como usuario

Esta función tiene como finalidad crear una cuenta de usuario. La secuencia temporal de la acción transcurre de la siguiente forma:

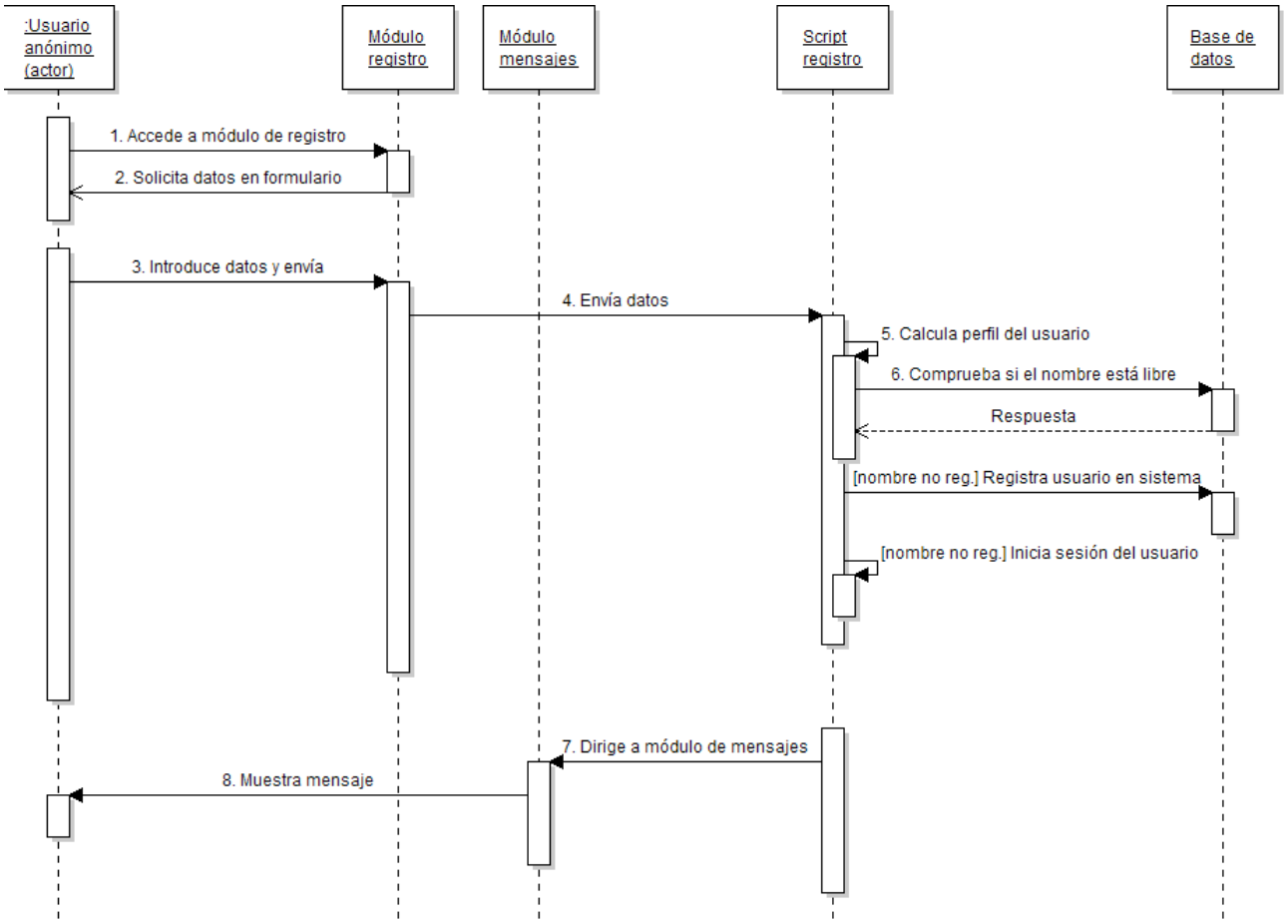

Figura 8.4 Diagrama de secuencia 1: registrarse como usuario.

El diagrama anterior muestra cómo transcurre el proceso de registro del usuario en el sistema. Un usuario anónimo accede a la página de registro, rellena un formulario con los datos solicitados y lo envía a la aplicación, que comprueba la disponibilidad del nombre de usuario introducido y en caso afirmativo lo registra en la base de datos del sistema, e inicia la sesión del nuevo usuario automáticamente. En caso de que ya existiera un usuario registrado con el mismo nombre, se avisará de tal circunstancia.

8.3.2 Diagrama de secuencia 2: iniciar sesión

La siguiente secuencia muestra cómo un usuario inicia sesión.

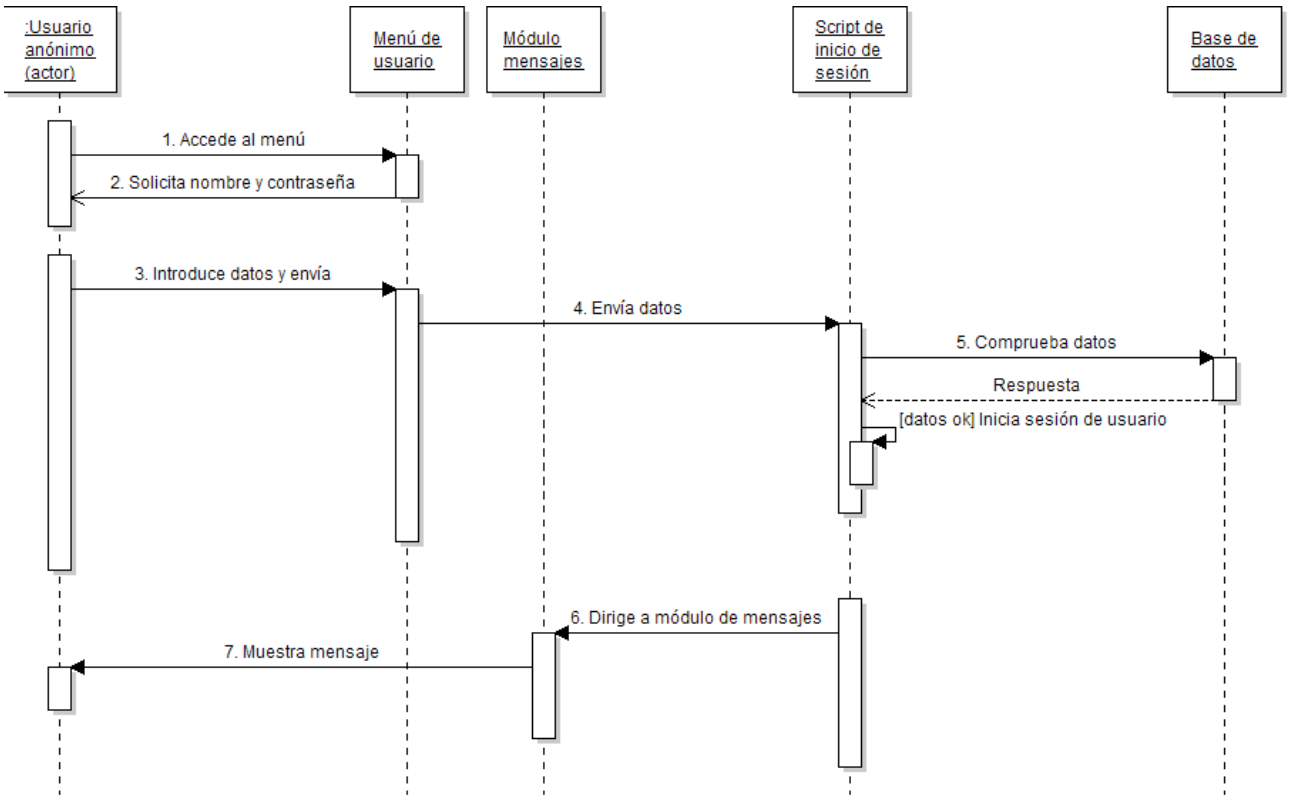

Figura 8.5 Diagrama de secuencia 2: iniciar sesión.

En este caso, un usuario anónimo previamente registrado que quiera iniciar sesión debe acudir al menú de usuario (accesible desde cualquier módulo), introducir su nombre de usuario y su contraseña, y enviar esos datos a la aplicación. la aplicación comprobará que el usuario se encuentra registrado en el sistema, que los datos introducidos son correctos. En caso afirmativo, iniciará la sesión del usuario y mostrará un mensaje. En caso contrario, mostrará un mensaje informando de que no ha podido iniciar dicha sesión.

8.3.3 Diagrama de secuencia 3: editar perfil

Cuando un usuario se registra, la aplicación calcula su perfil físico de acuerdo a los datos que ha introducido durante el proceso de registro. En cualquier momento, el usuario podrá volver calcular ese perfil si entiende que sus características y/o capacidades físicas han cambiado con el tiempo, tal y como se muestra en el siguiente diagrama:

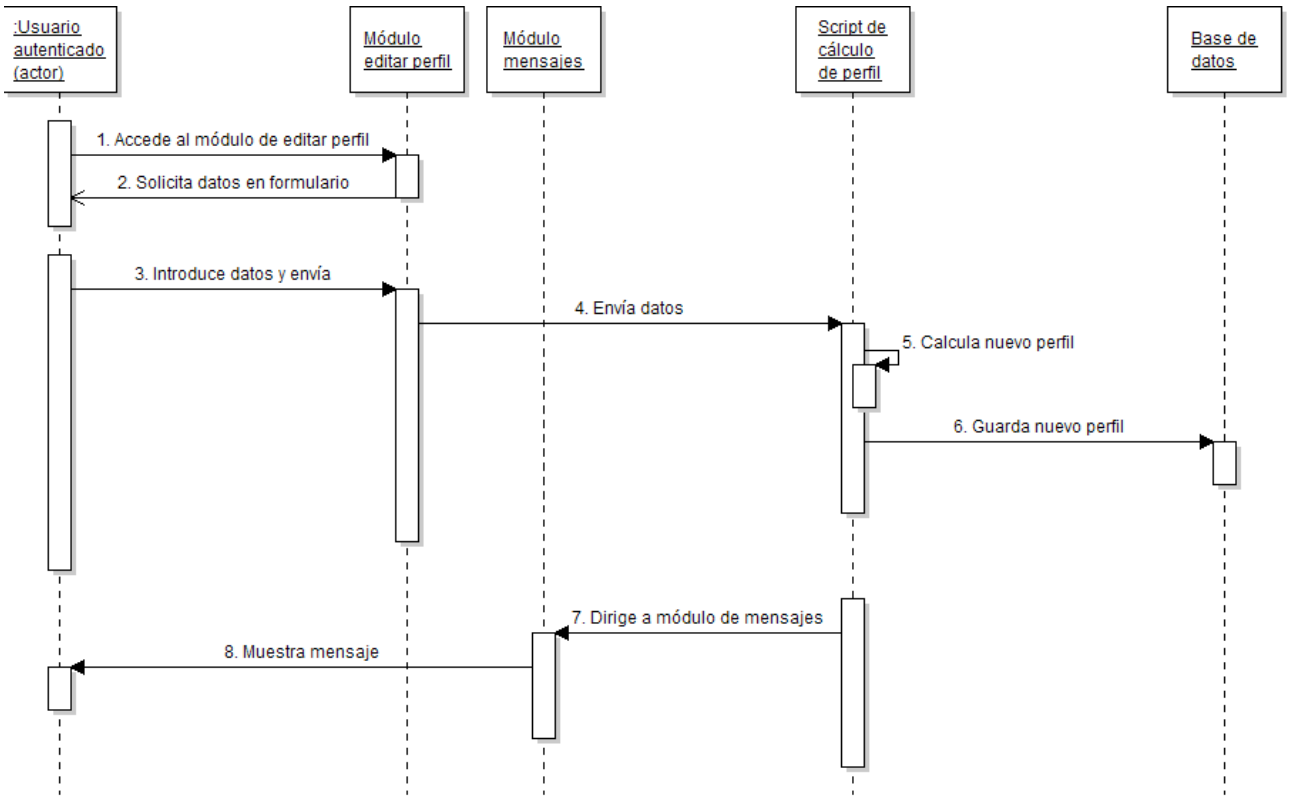

Figura 8.6 Diagrama de secuencia 3: editar perfil.

El usuario, que debe estar autenticado, accede al módulo de edición del perfil, rellena un formulario y envía los datos a la aplicación, que calculará el nuevo perfil, lo actualizará en la base de datos, y mostrará un mensaje al usuario confirmando la operación.

8.3.4 Diagrama de secuencia 4: editar datos personales

Esta operación es similar a la edición del perfil, y consiste en actualizar los datos personales del usuario, en concreto la contraseña y el correo electrónico.

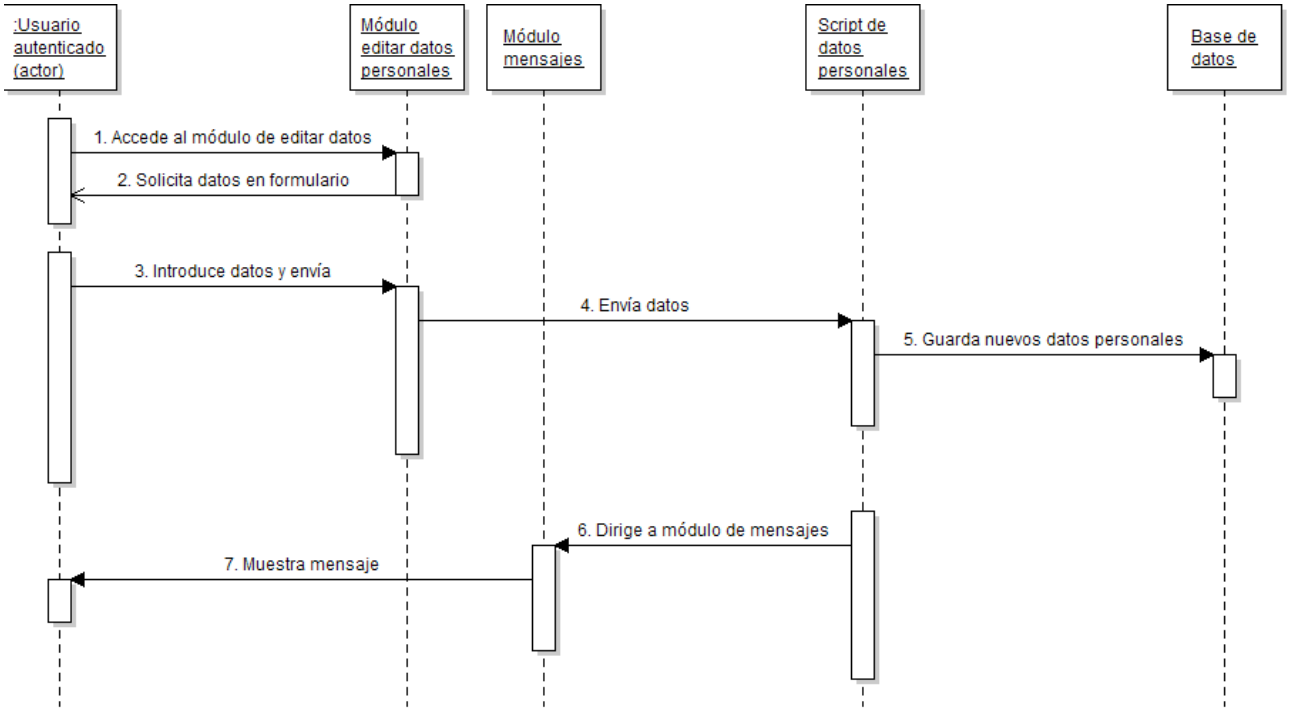

Figura 8.7 Diagrama de secuencia 4: editar datos personales.

8.3.5 Diagrama de secuencia 5: búsqueda de rutas recomendadas (usuario anónimo)

Los dos siguientes diagramas realizan la misma operación, pero los actores son distintos y no producen los mismos resultados. En este primer caso (Diagrama de secuencia 5), se trata de un usuario anónimo.

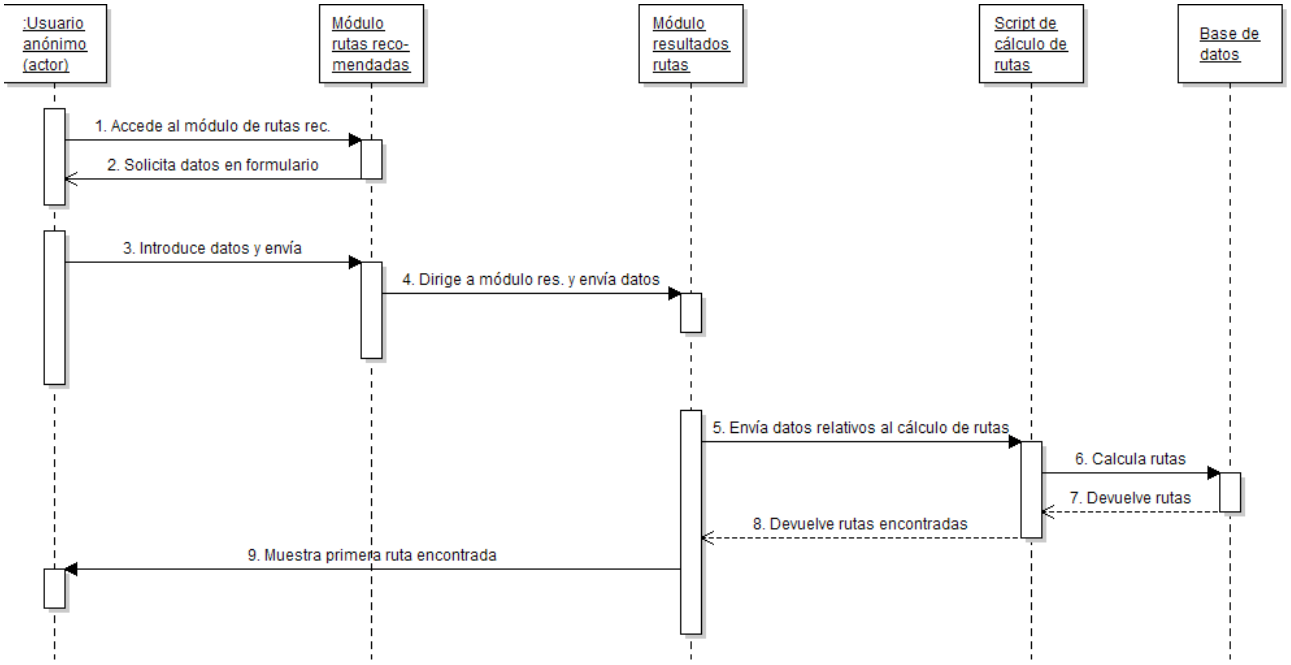

Figura 8.8. Diagrama de secuencia 5: búsqueda de rutas recomendadas (usuario anónimo).

El usuario accede en primer lugar al módulo de rutas recomendadas donde por medio de un formulario expresará qué rutas desea realizar. Cuando envía los datos del formulario a la aplicación, ésta calcula las rutas de acuerdo a esos datos y dirige al usuario al módulo donde se muestra el resultado de dicho cálculo. En este caso, por ser un usuario anónimo, sólo se mostrará un ruta, la primera que haya calculado la aplicación.

8.3.6 Diagrama de secuencia 6: búsqueda de rutas recomendadas (usuario autenticado)

Esta operación es prácticamente similar al diagrama de secuencia 5. El actor aquí es un usuario autenticado.

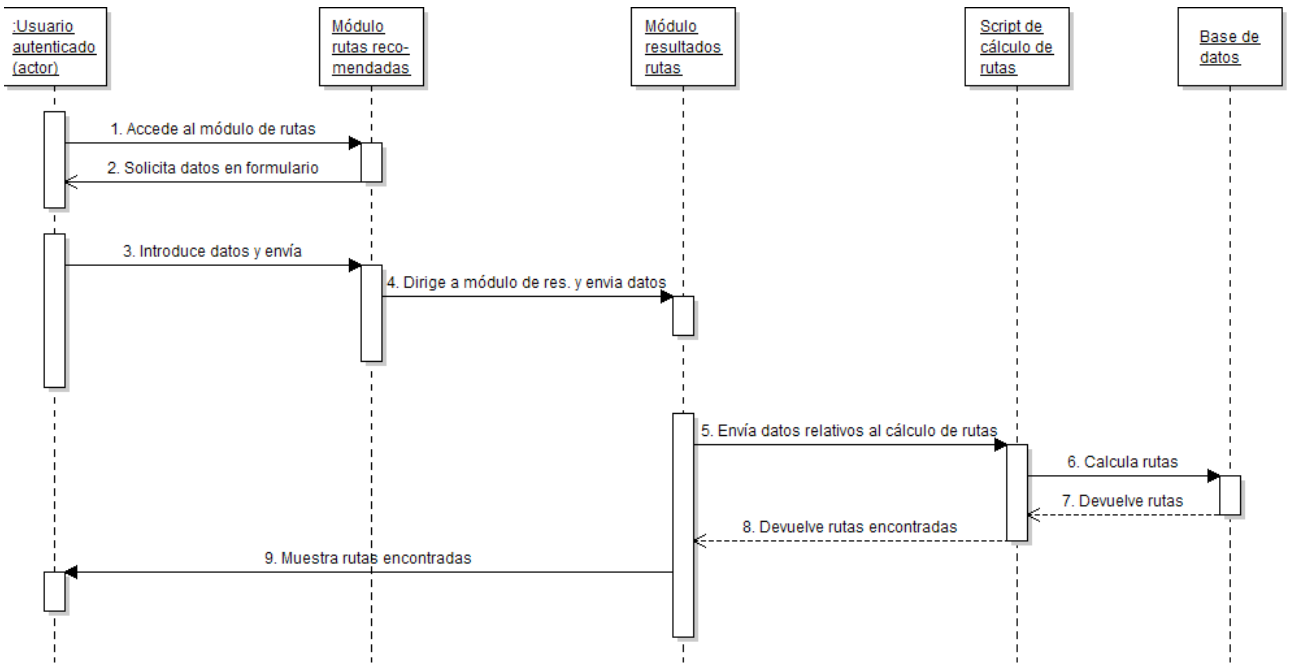

Figura 8.9. Diagrama de secuencia 6: búsqueda de rutas recomendadas (usuario autenticado).

El proceso es igual al descrito en el apartado anterior, salvo que en la página de resultados se muestran todas las rutas encontradas.

8.3.7 Diagrama de secuencia 7: búsqueda de la ruta más corta

El siguiente diagrama muestra el flujo de acontecimientos que suceden durante la acción de buscar la ruta más corta por parte de un usuario, ya sea anónimo o esté autenticado.

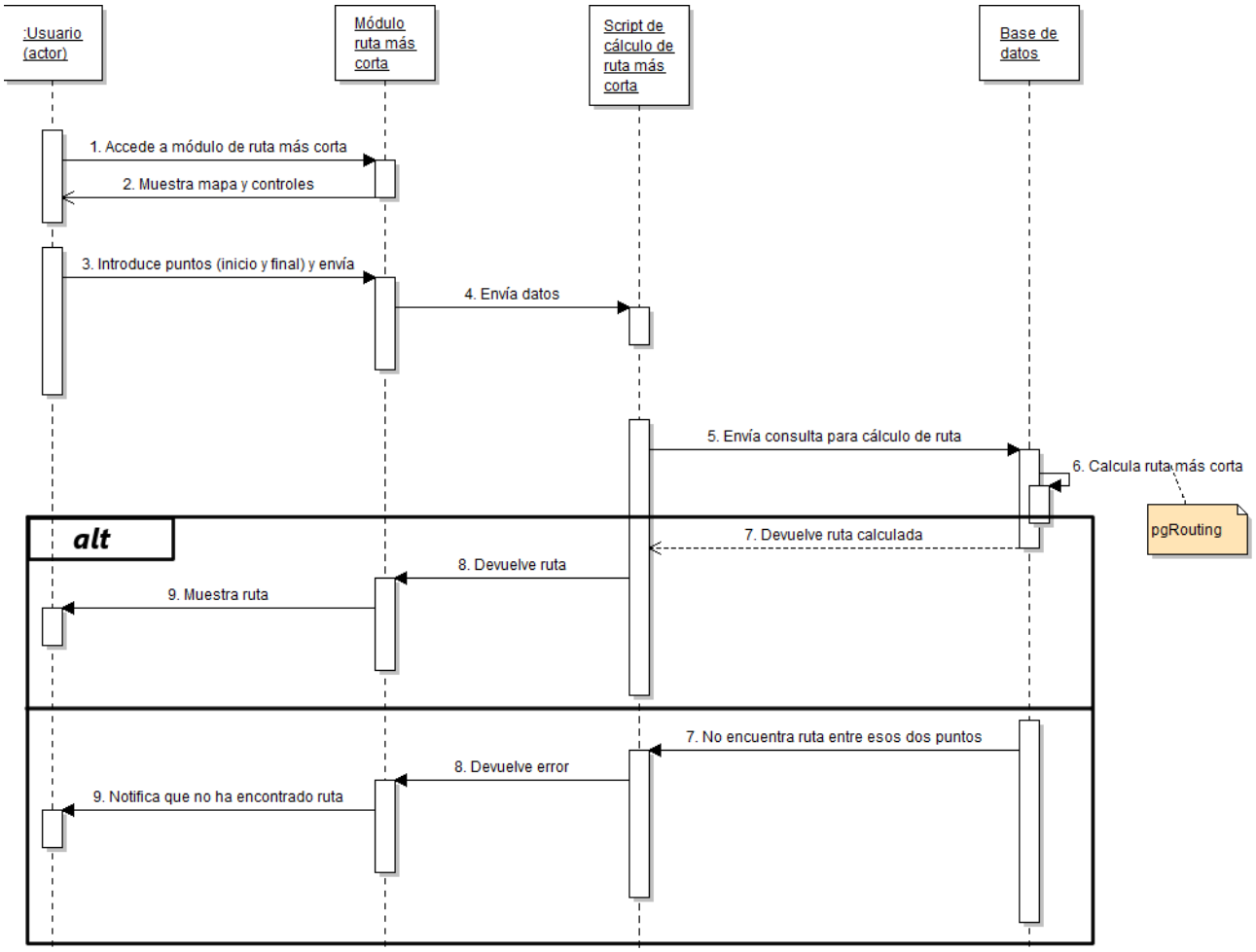

Figura 8.10. Diagrama de secuencia 7: búsqueda de la ruta más corta.

En este caso, el usuario accede al módulo de cálculo de ruta más corta, donde la aplicación muestra el mapa y unos controles para que el usuario pueda indicar los puntos de origen y destino de la ruta. El usuario envía los datos relativos a esos puntos y la aplicación trata de buscar la ruta más corta mediante los algoritmos implementados en la base de datos con pgRouting. Llegados a este punto pueden suceder dos cosas, tal y como queda reflejado en el diagrama:

- La aplicación ha encontrado la ruta más corta, y la muestra al usuario en el mapa del módulo de cálculo de ruta más corta.

- La aplicación no ha encontrado ninguna ruta entre esos dos puntos, y notifica al usuario de dicho suceso.

8.3.8 Diagrama de secuencia 8: agregar ruta

Esta acción puede ser realizada por un usuario autenticado, y consiste en añadir una ruta a la lista personal de rutas del usuario.

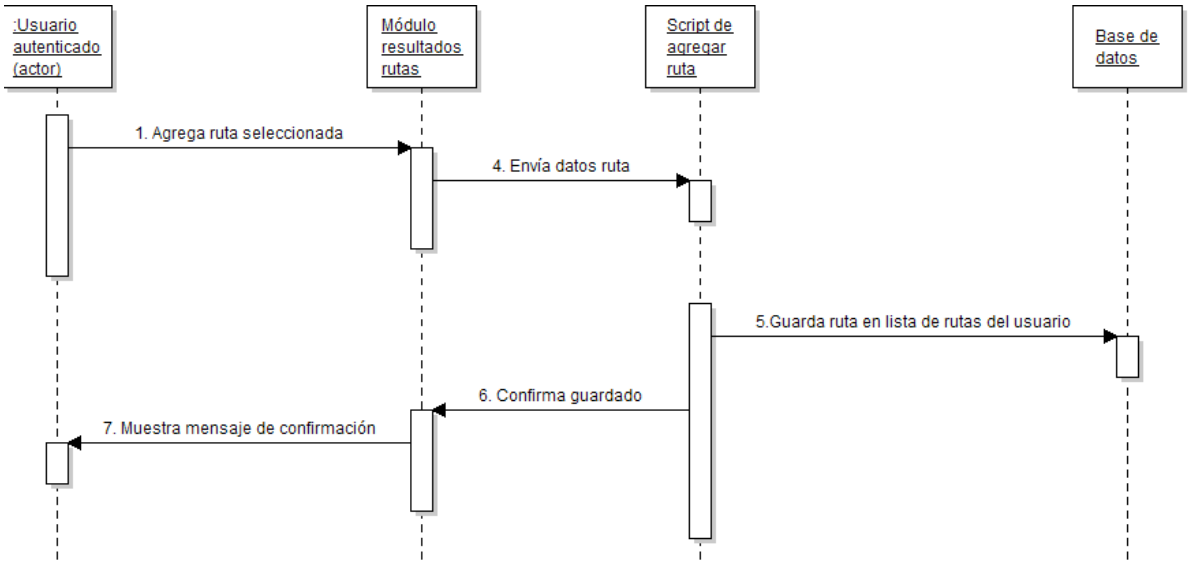

Figura 8.11. Diagrama de secuencia 8: agregar ruta.

El usuario puede realizar esta acción cuando se encuentra en el módulo de resultados del cálculo de rutas. Indica a la aplicación qué ruta quiere añadir, y ésta se encarga de guardarla en la lista personal de rutas del usuario en la base de datos.

8.3.8 Diagrama de secuencia 9: descargar ruta

Esta operación puede ser realizada por un usuario autenticado, y consiste en descargarse una ruta indicada por el usuario.

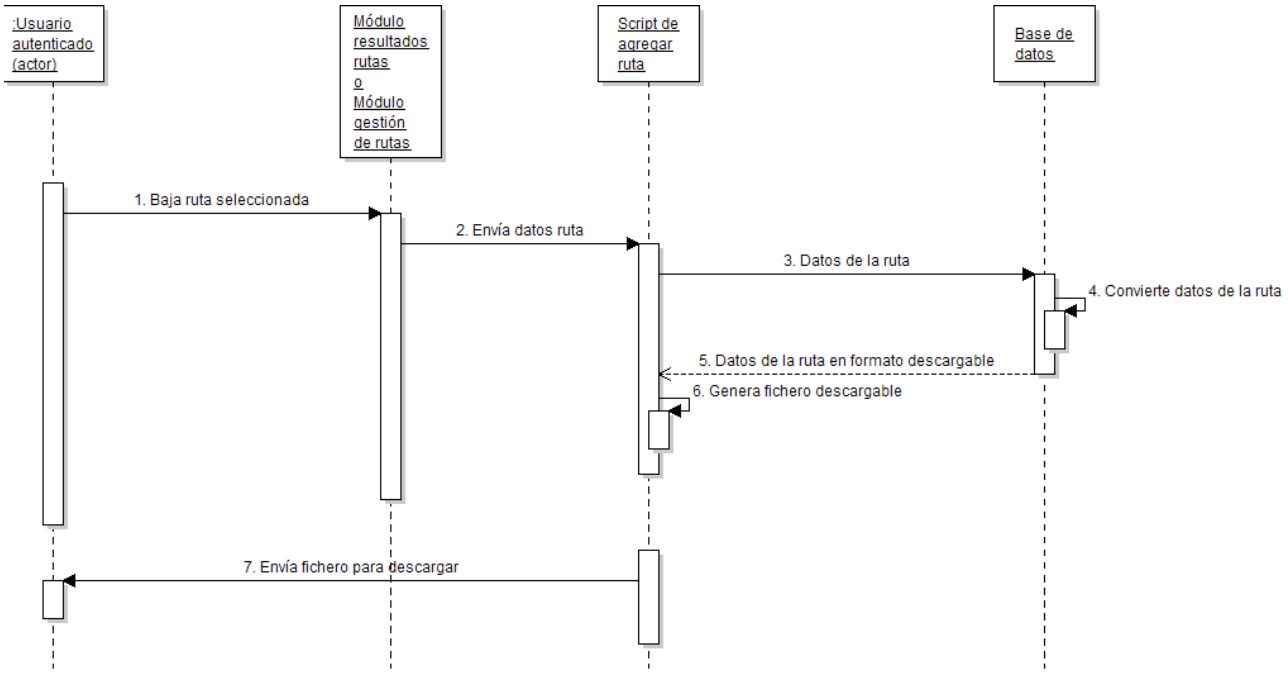

Figura 8.12. Diagrama de secuencia 9: descargar ruta.

El usuario puede realizar esta acción cuando se encuentra en el módulo de resultados del cálculo de rutas o en el módulo de gestión de rutas (las que tiene guardadas en su lista personal de rutas). Indica a la aplicación qué ruta quiere descargar, y ésta se encarga de transformarla en un fichero (mediante funciones de la base de datos PostGIS) y se la manda al usuario para que proceda a descargarla.

8.3.8 Diagrama de secuencia 10: borrar ruta

Esta acción puede ser realizada por un usuario autenticado, y consiste en borrar una ruta a la lista personal de rutas del usuario.

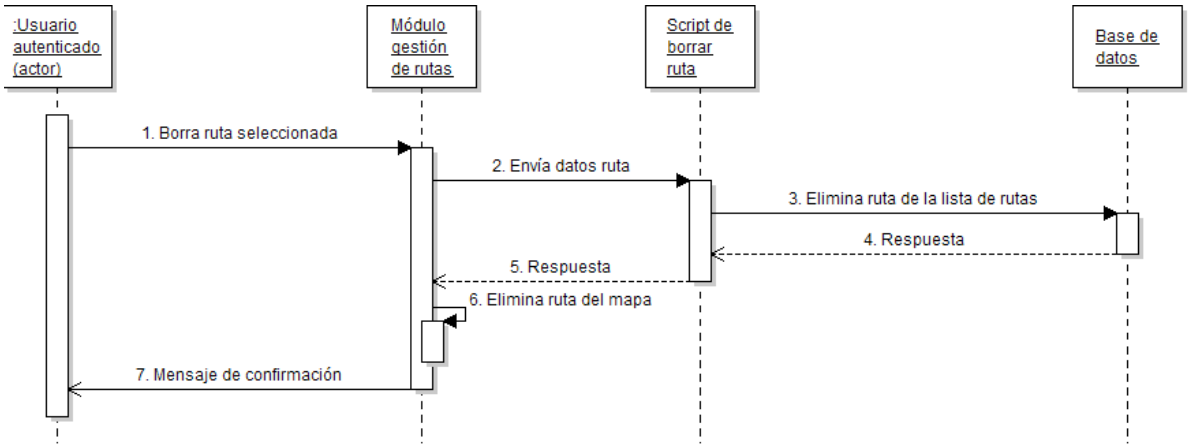

Figura 8.13. Diagrama de secuencia 10: borrar ruta.

Esta operación se realiza cuando el usuario se encuentra en el módulo de gestión de rutas (las que tiene guardadas en su lista personal de rutas). El usuario indica qué ruta desea borrar de su lista de rutas, y la aplicación la borra de la base de datos, mostrando a continuación un mensaje de confirmación.

8.3.11 Diagrama de secuencia 11: consultar usuarios del sistema

Esta operación sólo la puede realizar el administrador del sistema, y consiste en entrar en el módulo de panel de control del administrador, donde la aplicación mostrará una lista con los usuarios registrados en el sistema.

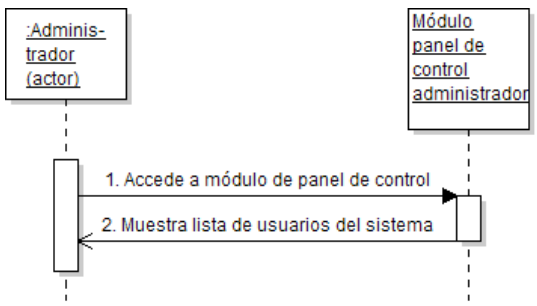

Figura 8.14. Diagrama de secuencia 11: consultar usuarios del sistema.

8.3.12 Diagrama de secuencia 12: borrar usuario

Dentro del módulo de panel de control del administrador, dicho administrador podrá borrar a un usuario del sistema.

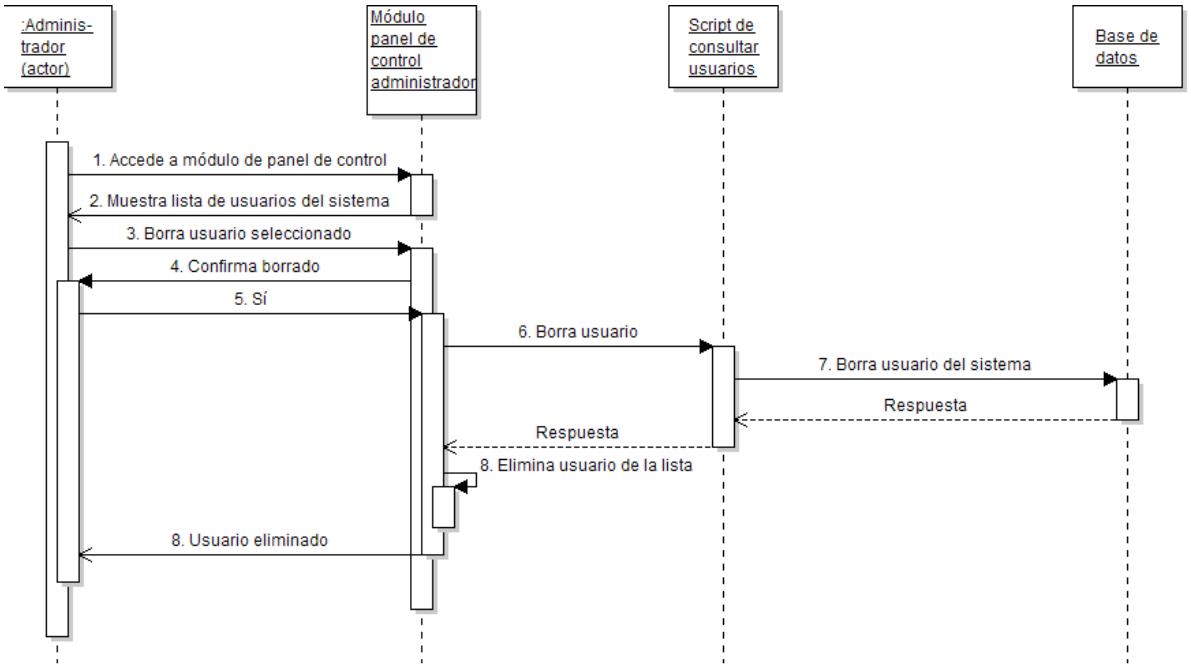

Figura 8.15. Diagrama de secuencia 12: borrar usuario del sistema.

El administrador accede a su panel de control, seleccionada un usuario de la lista e indica a la aplicación que desea eliminarlo del sistema. La aplicación se comunica con la base de datos y ésta elimina al usuario. Finalmente, la aplicación muestra un mensaje al administrador informando de ello.

CAPÍTULO

9 Análisis de la estructura de la información

En el presente capítulo se pretende describir tanto el tipo como la estructura de la información que manejará la aplicación.

9.1 Tipos de datos

La aplicación web va a poder consultar y generar los siguientes tipos de datos:

- Shapefile: formato multiarchivo de datos espaciales desarrollado por la compañía ESRI (Enviromental Systems Research Institute) [19]. Los shapefiles suministrados por el cliente son de dos tipos: vectorial (líneas, polígonos) y raster (imágenes).

- GeoTIFF: formato de imagen similar a TIFF que además permite almacenar metadatos con información georreferenciada [20]. Los datos raster incluyen datos en GeoTIFF, mientras que los Shapefile de esos datos raster se encargan de organizar esos GeoTIFF.

- PostGIS: los datos geoespaciales relativos a las rutas de la Sierra de las Nieves (datos vectoriales) serán almacenados en la base de datos PostGIS.

- GeoJSON: formato ligero para intercambio de datos entre cliente y servidor, es la variante geoespacial de JSON, y se usará para traer la información geográfica de la aplicación al cliente (en su navegador web) [21]. El cliente (navegador) realiza una petición mediante AJAX y recibe como respuesta datos en formato GeoJSON que el visor se encarga de manipular y transformar visualmente en rutas.

- KML: formato popularizado por Google Earth que está basado en XML, y sirve para almacenar y representar datos geográficos [22].

9.2 Estructura de la información

9.2.1 Shapefiles

El formato multiarchivo del Shapefile está compuesto por un archivo con extensión .shp (que almacena las entidades geométricas de los objetos), otro con extensión .shx (almacena el índice de dichas entidades geométricas), y otro con extensión .dbf (guarda la información de los atributos de los objetos). Opcionalmente también le puede acompañar otro archivo con extensión .prj, que guarda la información relativa al sistema de coordenadas.

9.2.2 GeoTIFF

Los archivos GeoTIFF pertenecientes a una capa se organizan en:

- Un directorio, que alberga archivos con las imágenes en formato TIFF y archivos de metadatos (con información general y geoespacial) de cada imagen.
- Un Shapefile que indexa y comunica a MapServer qué archivos TIFF debe leer en cada momento.

9.2.3 PostGIS

Los datos geoespaciales vectoriales almacenados en la base de datos PostGIS se guardarán en tablas, y cada una tendrá al menos un atributo (columna), que almacena la información geográfica: "the_geom", del tipo GEOMETRY. PostGIS permite manipular los datos geoespaciales como puntos, líneas, multilíneas, polígonos, etc.

La especificación usada para los datos almacenados en PostGIS es el estándar OpenGIS, que genera dos tablas (spatial_ref_sys y geometry_columns), donde se definen los sistemas de coordenadas espaciales y los tipos de geometría de los datos almacenados en la base de datos.

9.2.4 GeoJSON

El formato GeoJSON es una extesión del formato JSON, que proporciona un método ligero de intercambio de datos. La ventaja de GeoJSON sobre otros formatos es su simplicidad, rapidez e integración con la sintaxis de JavaScript, y permite especificar información relativa al sistema de referencia de los datos geoespaciales que acompaña de este formato.

Un ejemplo de objeto GeoJSON es el siguiente, donde guarda la información geográfica en un array "coordinates", indicando el tipo de geometría ("type":"MultiLineString"), o el sistema de referencia ("crs"):


```
{"longitud":1799.32429278,
"type":"FeatureCollection",
"features":[{"type":"Feature",
             "geometry":{"type":"MultiLineString",
                         "coordinates":[[[-4.97458978313,36.6789033882],
[-4.97569687544,36.6787719756],[-4.97662395803,36.67887116],
[-4.97719120286,36.6788044587],[-4.97786654575,36.6787932436],
[-4.97828941133,36.6786430082],[-4.9790068963,36.6788888661],
[-4.97936674125,36.679054738],[-4.97938361229,36.6797132277],
[-4.97942649191,36.6799989361],[-4.97975153127,36.6801940278],
[-4.97996039903,36.6800187024],[-4.98052691353,36.6799233547],
[-4.98062137858,36.6799577548],[-4.98066053572,36.68003567],
[-4.98060607806,36.6802371017],[-4.98086003026,36.6804333727],
[-4.98132872476,36.6806833545],[-4.98161528838,36.6807645128],
[-4.98190332157,36.6809029302],[-4.98215434192,36.6809846779],
[-4.98251567266,36.6812078003],[-4.98262671659,36.6813778041],
[-4.98341900741,36.6817656001],[-4.98413138928,36.6818110162],
[-4.98519554825,36.6817073531],[-4.98586649732,36.681524313],
[-4.98636413323,36.6815160128],[-4.98675807767,36.681624009],
[-4.98705054138,36.6819341922],[-4.98706158802,36.6823636393],
[-4.98699565161,36.6825652345],[-4.98686451421,36.6829970551],
[-4.98701185337,36.6831950905],[-4.98783972964,36.6835822626],
[-4.98834253581,36.6837743625],[-4.98880099116,36.6842756401]]]},
                         "crs":{"type":"EPSG",
                                "properties":{"code":"4326"}},
               "properties":{"id":null,"length":"0.0179932429277887"}}]}
```

9.2.5 KML

El formato KML define, con una sintaxis similar a XML, un título, una descripción textual básica de los datos, el tipo de datos (punto, linea, multilinea, etc.), las coordenadas (expresadas en longitud y latitud, y con el datum WGS84), e información adicional (como el estilo visual). Un ejemplo de KML es el siguiente:

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://earth.google.com/kml/2.1'>
<Document>
<name>'Caucón - Peña de los enamorados'</name>
<description>'Ruta por la Sierra de las Nieves'</description>

<Style id='defaultStyle'>
  <LineStyle>
    <color>55148EFF</color>
    <width>4</width>
  </LineStyle>
  <PolyStyle>
    <color>281400FF</color>
  </PolyStyle>
</Style>

<Placemark>
<styleUrl>#defaultStyle</styleUrl>
<MultiGeometry>
     <LineString><coordinates>--4.967813858280883,36.701864561555738
-4.967327827037412,36.701967985677626
-4.966827904876978,36.702107407585665
-4.966209528304873,36.704340236755407
-4.966142484891201,36.704498740015552
-4.966091504990966,36.704590203672517
-4.966029773003041,36.704724769907145
-4.965918488943328,36.704774304322001
     </coordinates></LineString>

     <LineString><coordinates>-4.965918488943328,36.704774304322001
-4.965771557526046,36.704819655602414
-4.965696405989121,36.70489244062415
-4.965650496749422,36.704950433231495
-4.965670079732863,36.705021652220303
-4.965765788154576,36.705058228897144
-4.965944753118094,36.705107738398446 -4.966093263714632,
     </coordinates></LineString>
</MultiGeometry>
</Placemark>

</Document>
</kml>
```

CAPÍTULO

10 Análisis del interfaz web

En el presente capítulo se analizará la conexión entre el sistema y el usuario final, es decir, el interfaz de usuario. En primer lugar se definirá qué tipos o perfiles de usuario interactuarán mediante dicho interfaz con la aplicación, y a continuación se realizarán unas consideraciones previas al diseño del interfaz.

10.1 Perfiles de usuario

La aplicación a desarrollar ofrecerá la posibilidad de crear una cuenta de usuario en ella. Diferenciará entre tres perfiles de usuario:

- Anónimo: es el usuario por defecto, que no ha iniciado sesión y tiene el acceso limitado a algunas funciones de la aplicación, como por ejemplo descargar rutas o gestionar una lista personal de rutas.

- Autenticado: usuario que previamente se ha registrado en la aplicación y ha iniciado sesión, tiene acceso completo a todas las funciones de la aplicación, salvo las referentes al siguiente perfil, el administrador.

- Administrador: usuario autenticado que además puede realizar tareas adicionales de administración del sitio, como gestionar los usuarios registrados.

10.2 Consideraciones previas al diseño de la aplicación

La aplicación web se desarrollará como un portal web, y en esta fase hay que tener definida la estructura de programación y la estructura visual de la web, para abordar con garantías el posterior desarrollo de la misma.

10.2.1 El modelo de las tres capas

Para implementar el código de la aplicación web se seguirá el modelo de separación en tres capas diferenciadas según su significado [23], muy similar al Modelo Vista Controlador (MVC), y cada una de ellas hará uso de un lenguaje de programación:

- Capa de estructura o contenido: "qué significa", describe el contenido de la página. Esta capa estará escrita en XHTML (eXtensible HyperText Markup Language) 1.0, la versión semántica del tradicional HTML. Con semántico, quiere decir que no definimos el aspecto de las cosas, sino lo que significan.

- Capa de presentación: "cómo se muestra", especifica la presentación de ese contenido. Se programará en CSS (Cascading Style Sheets) en su versión 2, con algunas funcionalidades de la versión 3.

- Capa de comportamiento: "qué hace", controla el comportamiento de ese contenido. Estará escrita en JavaScript (también conocido como ECMAScript).

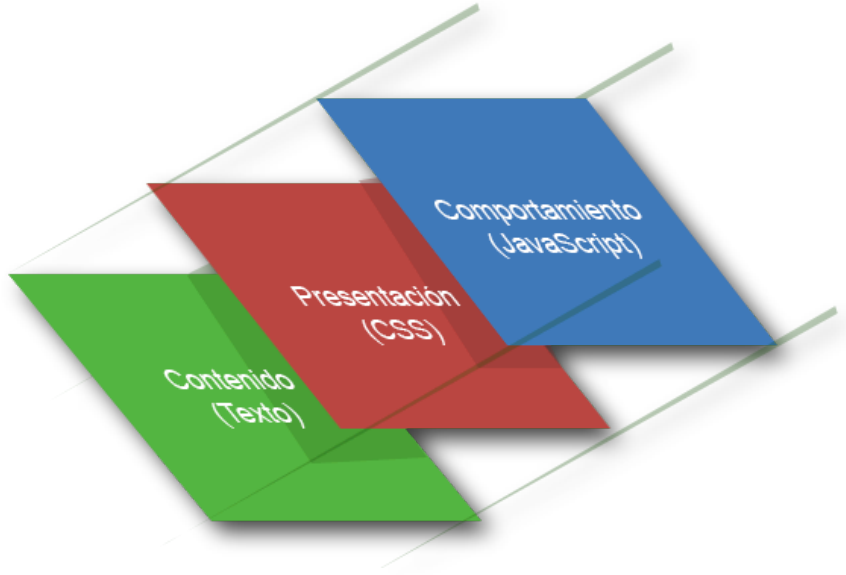

Figura 10.1. Modelo de las tres capas.

De esta forma, el proceso de desarrollo también fluye por estas tres capas a modo de etapas, ya que:

- Se empieza produciendo el contenido en formato XHTML. Esta es la capa base, y cualquier usuario con cualquier navegador web deberá poder visualizarla.

- Con esta capa definida, los esfuerzos se pueden centrar a continuación en dotar a la página web de un estilo visual más atractivo, añadiendo una capa de presentación con información escrita en CSS. El sitio web lucirá mejor para los usuarios cuyos navegadores puedan mostrar estilos CSS.

• Por último, se puede usar JavaScript para introducir una nueva capa de interactividad y comportamiento dinámico, que dotará al sitio web de nuevas funcionalidades y una mayor sencillez en el manejo en los navegadores que sean capaces de ejecutar JavaScript.

Con esto no sólo se ofrece una forma ordenada de diseño y posterior mantenimiento al desarrollador, sino que abarca el mayor rango posible de visitantes del sitio web, ya que se ejecutará (aunque de forma limitada) en aquellos navegadores que no soporten las capas superiores (la capa base la podrán leer todos los navegadores).

En este sentido, atendiendo a los lenguajes de programación y tecnologías que vamos a usar, hay que considerar lo siguiente:

• El lenguaje XHTML y HTML 4.01 están establecidos actualmente. En el horizonte asoman HTML5 y XHTML5, que todavía se encuentran en desarrollo y por esa razón no se usan en este proyecto, aunque los navegadores más modernos ya lo soportan.

• El lenguaje CSS 2 también avanza (la versión actual es 2.1) y ya casi está CSS 3, cuyo camino empezó en 1999 y que se encuentra en fase de borrador final por parte del consorcio W3C (organismo encargado de desarrollar especificaciones estándar de la Web), aunque la mayoría de navegadores son compatibles con esta nueva versión. Añade nuevas funcionalidades manteniendo todo lo anterior, así que en futuras actualizaciones se podrán modificar o añadir diferentes características de forma sencilla sin alterar todo lo que hay hecho.

• JavaScript (o ECMAScript) ha ido evolucionando de forma tortuosa desde que fuera creado allá en 1995. Netscape delegó en la ECMA (European Computer Manufacturers Association) la normalización del lenguaje, y lo desarrolló bajo el nombre de JavaScript, mientras que Microsoft decidió crear su propia versión, denominada jscript. Las sucesivas actualizaciones de estos lenguajes han ido convergiendo poco a poco hacia el estándar ECMAScript, aunque no de forma uniforme en todos los casos. Pero en las nuevas versiones de los navegadores, incluido el de Microsoft, se puede afirmar que ECMAScript ha superado casi al completo esta fase de incompatibilidades y su funcionamiento es independiente del navegador que estemos usando.

• La librería de JavaScript jQuery está muy extendida y apoya la filosofía de *cross-browser*, por lo que es compatible con todos los navegadores.

• El lenguaje del lado del servidor será PHP, que goza de mucha popularidad en el desarrollo web, es multiplataforma, es libre y está en continua evolución. También es muy flexible en cuanto a la metodología de programación que se desee seguir, por lo que se puede adaptar facilmente a nuevos escenarios.

Las principales directrices que hay que seguir a la hora de programar en XHTML (y que en algunos casos lo diferencia de HTML) son las siguientes:

• La primera línea de un documento XHTML debe marca la codificación de caracteres (formato en el que se guardan). De acuerdo con las recomendaciones del consorcio W3C, se usará la codificación Unicode UTF-8, que soporta caracteres de todas las lenguas, y permite traducir la página a cualquier idioma.

• También es conveniente indicar el tipo de documento con la directiva DOCTYPE, en este caso va a ser XHTML 1.0 Transitional.

• La estructura guardará un cierto orden, y así por ejemplo en la cabecera (etiqueta <head>) se incluirá los vínculos a hojas de estilo CSS, scripts, información para robots de búsqueda, o el título del documento, por ejemplo. En el cuerpo del documento (etiqueta <body>) no habrá ningún script (etiqueta <script>).

• Las etiquetas pueden tener la forma <etiqueta>Contenido</etiqueta>, o funcionar con una sola parte, <etiqueta />. En cualquier caso siempre hay que cerrarlas con la barra '/', y en el segundo caso es conveniente dejar una espacio en blanco antes de la barra, para que los navegadores antiguos lo entiendan.

• Las etiquetas deben ir siempre en minúsculas, y además pueden llevar atributos, que siempre deben ir entre comillas dobles. Ejemplo: <etiqueta atributo="valor">...</etiqueta> .

• Todas las imágenes (etiqueta <img>) incluirán el atributo 'alt', que contiene una descripción de la imagen, necesaria cuando por algún motivo no se haya podido cargar la imagen, y también por motivos de accesibilidad.

• No se usarán tablas para maquetar las páginas web. Antes de la llegada de CSS no había otra forma de maquetar webs (salvo usar *frames*, algo poco aconsejable). Pero con CSS se hará por medio de capas, cuyo aspecto, tamaño y ubicación se definen en la hoja de estilos, y así no habrá necesidad de modificar código HTML.

• Las etiquetas de los campos de los formularios (etiqueta <form>) llevarán un texto asociado mediante la etiqueta <label>, por razones de accesibilidad.

Y en CSS:

• Se evitará el uso de *hacks* (trucos para visualizar características de CSS en navegadores poco "exigentes" con los estándares W3C), siempre que no se entre en conflicto con el objetivo de ser *cross-browser*.

• Se añadirá al principio una hoja de estilo que reinicie todos los valores que suelen llevar por defecto (y no siempre coincidiendo) todos los navegadores, para que la página se visualice igual en cualquier navegador.

10.2.2 Diseño modular de la aplicación

De forma complementaria al diseño de las tres capas, la implementación de la aplicación web se abordará mediante este paradigma de la programación, que consiste en dividir el programa en módulos con el fin de distribuir la complejidad en varios problemas (subprogramas) más sencillos, hacerlo más legible, aumentar la escalabilidad y facilitar el posterior mantenimiento del mismo.

El lenguaje PHP facilitará enormemente esta tarea [24], ya que permite incluir una serie de mecanismos enfocados a un diseño modular estándar, como por ejemplo añadir código que resida en un archivo separado directamente en documentos HTML (mediante el procedimiento *include("nombre_archivo")*), algo parecido a lo que realizan lenguajes como C++.

Los objetivos que se desean alcanzar con el uso de este modelo en PHP, aparte de los ya comentados, son:

• Diseñar y centralizar el proceso de creación y configuración de los módulos. Por ejemplo, con un archivo de configuración que defina los módulos existentes en el sitio web, junto a unos parámetros básicos (como el directorio donde se encuentra el archivo, o el nombre del mismo). Esto será la clave de la escalabilidad, ya que para añadir un nuevo módulo sólo habrá que editar este archivo de configuración.

• Diseñar y centralizar el proceso de carga de módulos, dado que el navegador cada vez que necesite solicitar una página va a realizar una petición basada en un sólo dato: la dirección URL. A esa dirección se le pueden añadir unos parámetros que especifiquen el módulo que se desea cargar, entre otras cosas.

- Flexibilizar la estructura de la web (XHTML), mediante el uso de plantillas. Esto facilitará cambios en el diseño, si el cliente así lo desea.

- Organizar los archivos en directorios.

- Reducir el acoplamiento y aumentar la independencia del código, para facilitar su modificación.

En el capítulo de implementación de la aplicación web se explicará más a fondo cómo se han llevado a cabo estos objetivos.

10.2.3 Diseño de la interfaz web cliente

Con el modelo de las tres capas y el diseño modular en mente, la tarea de diseñar la interfaz de la aplicación web se puede afrontar, partiendo de unos requisitos básicos solicitados por el cliente, con cierta libertad por parte del desarrollador, y garantías de rediseño en caso de que el cliente crea necesario modificar algo del diseño original.

Las posibilidades que ofrecen tanto CSS como algunas librerías de JavaScript (jQuery) son enormes, y permiten dotar a la aplicación web de un gran atractivo visual. La misión del desarrollador deber ser encontrar el equilibrio entre todos los aspectos que están en juego, es decir, crear un aspecto visual que sea atractivo pero no excesivamente recargado para no entorpecer la claridad y legibilidad del contenido.

La aplicación tendrá una primera página de presentación con un estilo visual diferenciado, con enlaces a la página principal, al buscador de rutas y a la información general de la Sierra de las Nieves, así como una sección "Acerca de la web" que comentará brevemente el propósito y la utilidad de la misma.

La página principal tendrá el siguiente diseño, que seguirá un patrón similar en el resto de secciones (menú superior del usuario, menú horizontal de navegación justo antes del contenido de la sección, etc.):

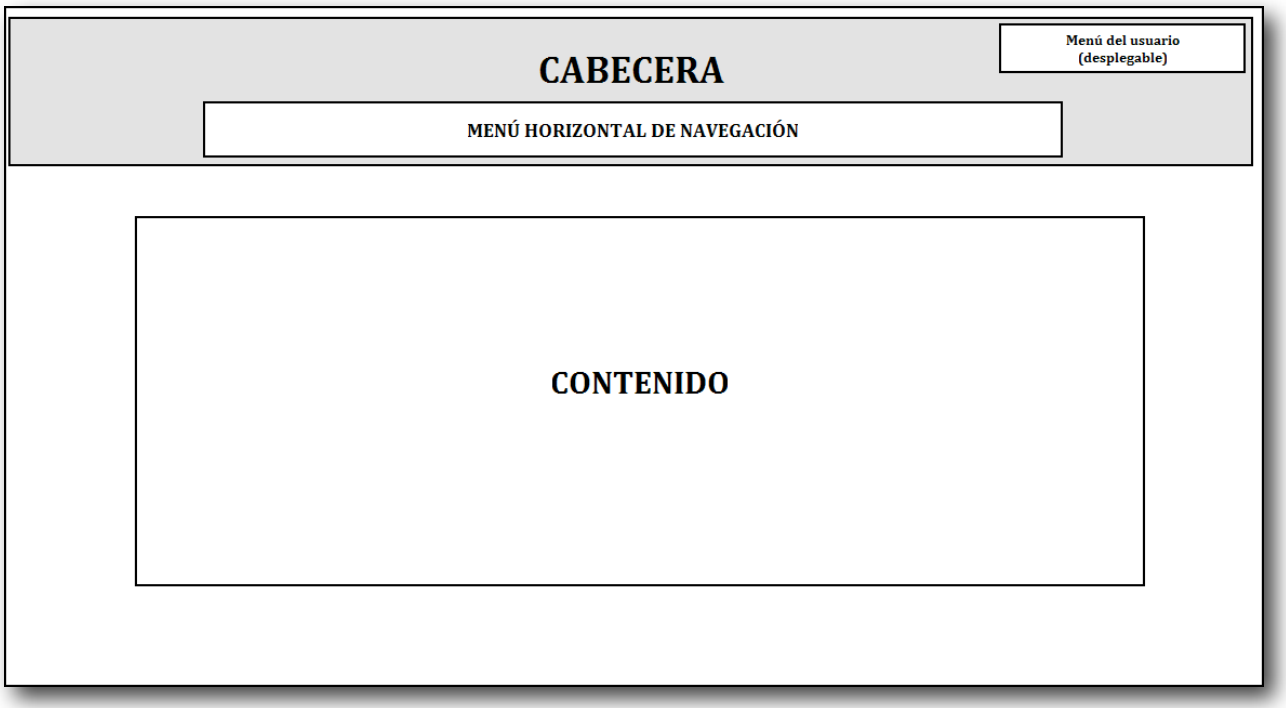

Figura 10.2. Diseño del interfaz de la página principal.

Esta será la plantilla general para el resto de páginas. La página que muestre los resultados de la búsqueda de rutas, por ejemplo, tendrá el siguiente aspecto:

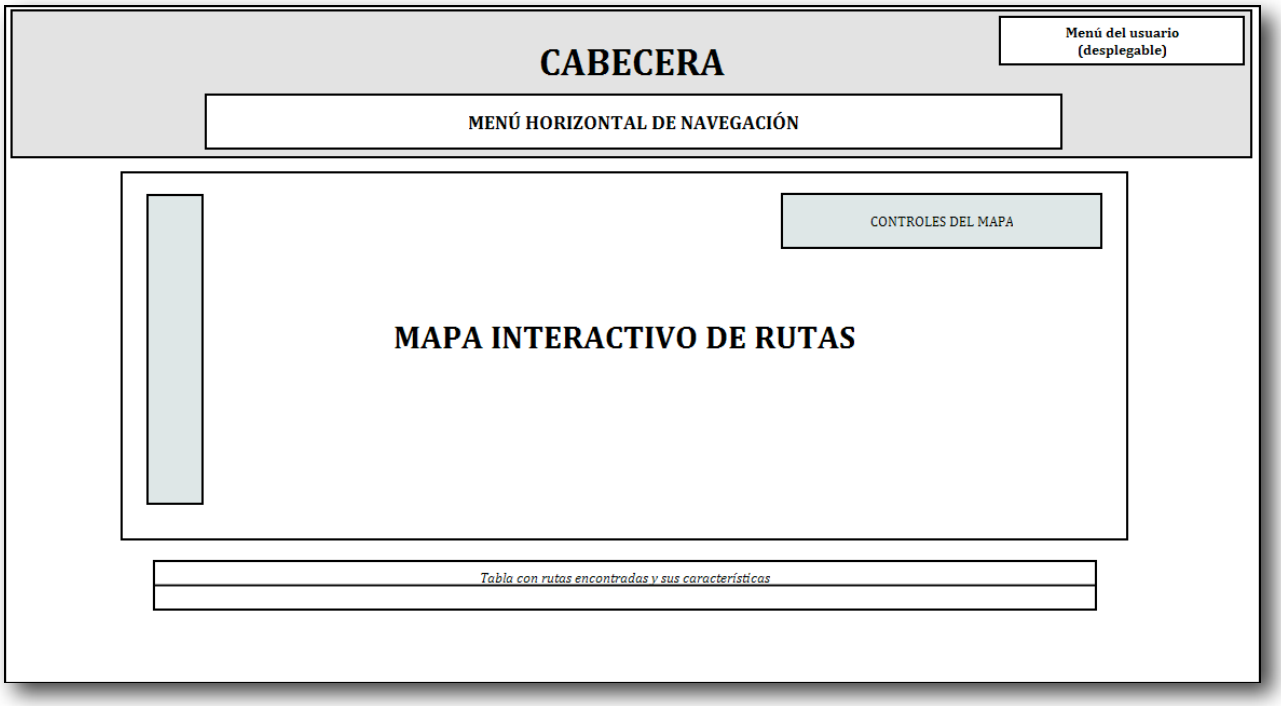

Figura 10.3. Diseño del interfaz de la página de resultado de rutas.

La plantilla para la página de presentación difiere de esta estructura y es más simple. Mostrará el siguiente diseño:

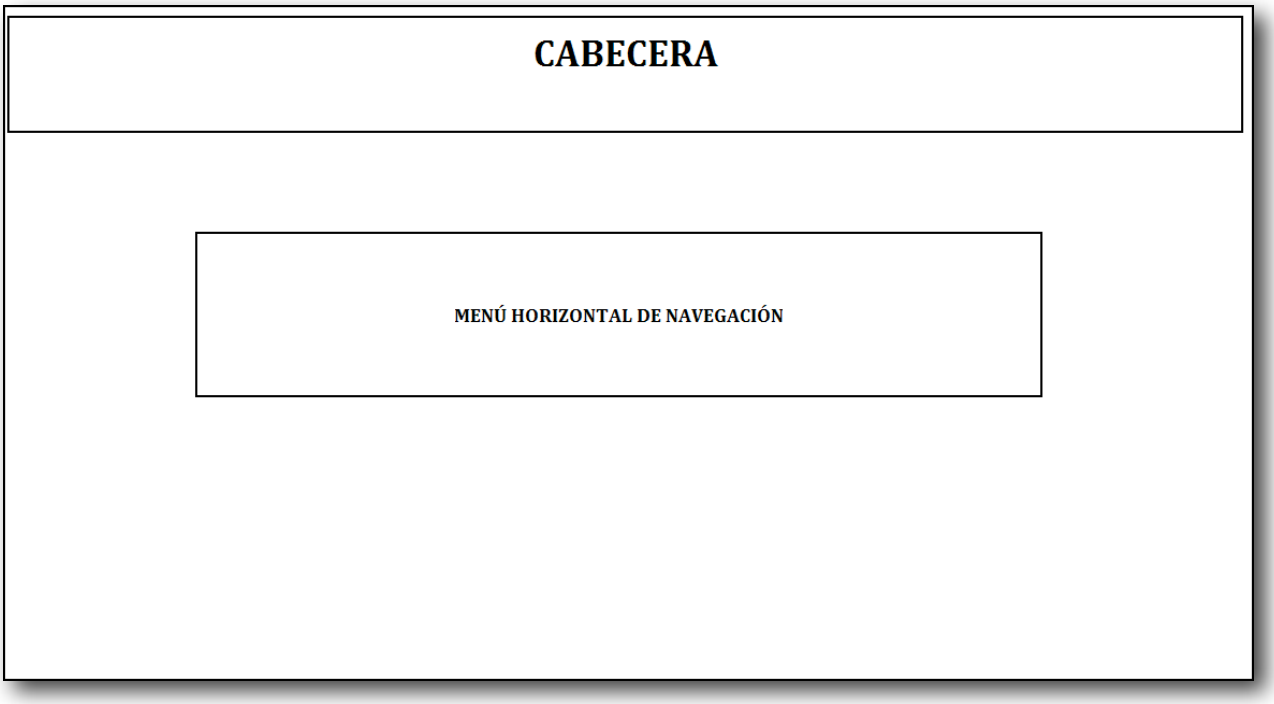

Figura 10.4. Diseño del interfaz de la página de presentación.

BLOQUE 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

CAPÍTULO

11 Instalación de la IDE

Una vez trazado el esquema que debe seguir nuestra IDE, el siguiente paso es implementarlo en el servidor. Para ello se van a seguir una serie de pasos.

11.1 Instalación de componentes en el servidor

La primera tarea consiste en instalar los siguientes componentes en el servidor y comprobar su correcto funcionamiento (la instalación paso a paso se encuentra en el ANEXO I de este documento):

- El servidor web HTTP Apache.
- El servidor de mapas UMN MapServer.
- La base de datos PostgreSQL junto a la extensión PostGIS y la librería pgRouting.

11.2 Datos geoespaciales

En segundo lugar hay que añadir los datos geoespaciales a la IDE. En función del formato de los archivos, del uso que se pretenda hacer de ellos y de la naturaleza de los datos geográficos se realizarán las respectivas operaciones.

11.2.1 Shapefile

En el caso de los archivos en formato Shapefile, tan sólo hay que crear un directorio donde almacenarlos, y MapServer podrá leerlos e interpretarlos directamente, indicándole que acceda a ese directorio. En el siguiente apartado se explicará cómo lleva a cabo ese trabajo.

11.2.2 Datos en PostGIS

Los datos susceptibles de ser analizados y tratados por la aplicación web (fundamentalmente los que describen los senderos del Parque de la Sierra de las Nieves) deben ser volcados a la base de datos PostGIS para que, entre otras cosas, se puedan realizar cálculos de la ruta más corta (con la librería pgRouting) sobre ellos, o almacenar rutas a petición de los usuarios de la web.

Lo primero que hay que hacer es crear una base de datos PostGIS con todas las funciones de pgRouting, donde se almacenará la información geoespacial expuesta. En el ANEXO I de esta memoria se detalla el proceso para crear dicha base de datos.

Los datos de partida están en formato Shapefile, así que es necesario realizar algún tipo de transformación, dado que PostGIS es una base de datos que no puede leer directamente los Shapefile. La herramienta Shp2pgsql soluciona este problema, ya que es capaz de convertir los Shapefiles en sentencias SQL listas para ser insertadas en una base de datos de PostgreSQL/PostGIS tanto en formato geométrico como geográfico.

11.2.3 Cambio del sistema de coordenadas y proyección

Los datos de partida proporcionados por el cliente se encuentran en formato ED50, y requiere una configuración propia, descrita en el ANEXO I de este documento (concretamente en el capítulo 19).

11.3 Configuración de UMN MapServer

El último paso para tener la IDE completamente operativa es configurar MapServer, que como se ha visto en el capítulo 9 supone editar el archivo Mapfile. El Mapfile contiene todas las capas que se quieren mostrar en la IDE, así como cierta información general de configuración de MapServer. A continuación se mostrarán las partes de código relacionadas con la puesta en marcha de la IDE.

El servicio de mapas que proporciona MapServer a través de la IDE se hará mediante el protocolo Web Map Service (WMS) 1.1.1. A la hora de configurar el Mapfile es indispensable indicar lo siguiente en la definición del mapa:

- Nombre del mapa, como se muestra a continuación:

```
NAME “sierranieves”
```

• Los metadatos definidos y requeridos por el protocolo WMS:

```
WEB
      METADATA
         "wms_encoding" "UTF-8"
         "wms_title" "Mapserver Sierra de las Nieves"
         "wms_srs" "EPSG:900913 EPSG:4326"
         "wms_onlineresource" "http://localhost/cgi-bin/mapserv?map=/
home/user/mapa.map"
      END
END
```

• La proyección de los datos. Si alguna capa del Mapfile está en otra proyección distinta, se puede definir esa proyección de la misma forma dentro de la capa.

```
PROJECTION
      "init=epsg:900913"
END
```

Si se desea profundizar más en las opciones de configuración de MapServer, el código completo del Mapfile se encuentra en el ANEXO II de este documento.

CAPÍTULO

12 Diseño de la aplicación

En este capítulo se mostrará la implementación de la aplicación que servirá como interfaz web entre el usuario y la IDE diseñada. Se partirá de los diseños modular y de tres capas descritos en capítulos anteriores, y se detallará todo el proceso de programación, los problemas y las soluciones que vayan surgiendo.

12.1 Web modular

En el apartado 10.2.2 se expusieron una serie de objetivos a alcanzar durante el desarrollo de la aplicación web siguiendo el patrón de diseño modular:

- Diseñar y centralizar el proceso de creación y configuración de los módulos.
- Diseñar y centralizar el proceso de carga de módulos.
- Flexibilizar la estructura de la web (XHTML), mediante el uso de plantillas.
- Organizar los archivos en directorios.
- Encapsular y repetir lo menos posible el código.

A continuación se muestra de qué forma se han alcanzado.

12.1.2 El archivo conf.php

Se trata del archivo de configuración de módulos, que se encarga de:

• Definir una serie de constantes que se van a utilizar a lo largo de la aplicación, referidas a nombres de archivo y rutas de directorios.

```
define('HEAD_DEFECTO', 'head.html');
define('MODULO_PATH', realpath('./modulos/'));
```

• Definir mediante un array asociativo (almacenado en la variable *$conf*) los módulos que componen la aplicación, así como qué plantilla (*layout*) usa cada uno, qué cabecera (*head*), el tipo de usuario (tiene que ver con el control de usuarios, que se explicará en un apartado posterior), y el título.

```
$conf['home'] = array(
             'archivo' => 'home.php',
             'layout' => LAYOUT_DEFECTO,
             'head' => 'headhome.html',
             'tipousuario' => 'todos',
              'titulo' => 'Sierra de las Nieves :: Inicio');"
```

12.1.2 El archivo index.php

El archivo principal (*index.php*) carga los módulos según el parámetro que se le pase (vía dirección URL), y el resto de datos que figuren en el elemento del array asociativo *$conf* (descrito en el apartado previo) cuyo nombre. Si no se le pasa argumentos carga por defecto la página principal, como se puede ver en el siguiente fragmento de código:

```
/** Verificamos que se haya escogido un modulo, si no
* tomamos el valor por defecto de la configuración.
*/
if (!empty($_GET['mod']))
       $modulo = $_GET['mod'];
else
       $modulo = MODULO_DEFECTO;
```

De igual forma, carga la plantilla correspondiente al módulo. Otra función que realiza, relacionada con la seguridad, es controlar el tipo de usuario que accede al módulo.

El código completo tanto de *index.php* como de *conf.php* se encuentra en el ANEXO II de este documento.

12.1.3 Plantillas

Las plantillas nos permiten definir la estructura del documento y están escritas en XHTML. Todas comenzarán definiendo el tipo de documento (mediante la directiva <!DOCTYPE), en este caso va a ser un documento XHTML 1.0 Transitional, y a continuación se estructurarán siguiendo la síntaxis definida por XHTML:

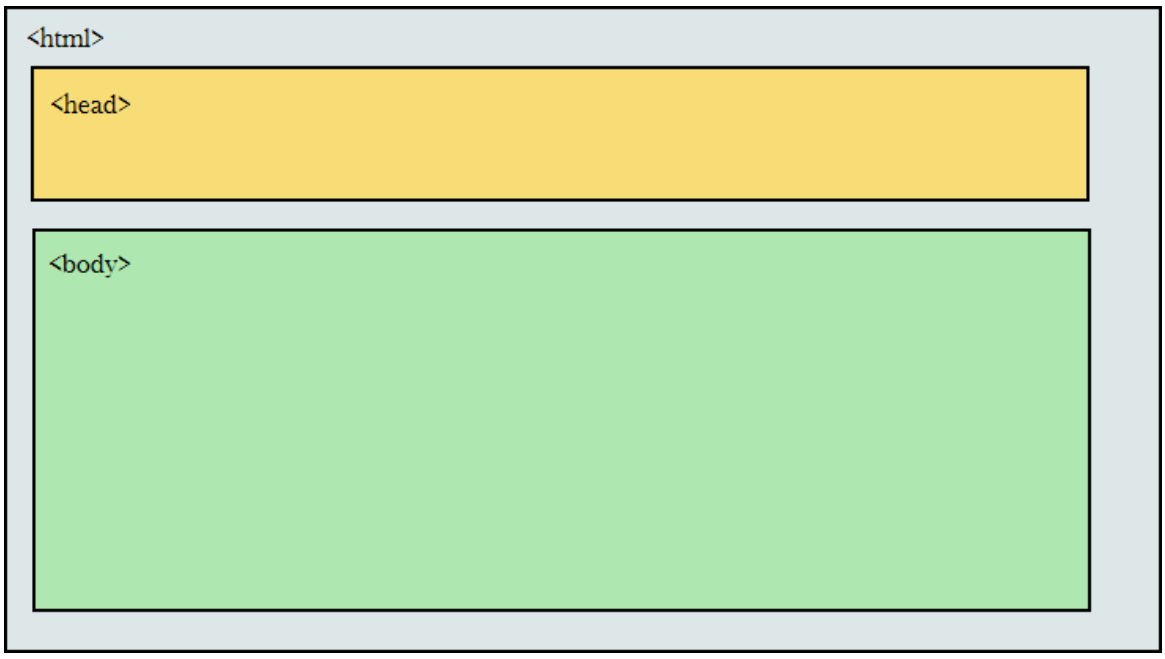

Figura 12.1. Estructura básica de XHTML.

En la cabecera (<head>) se ubicarán el título de documento, los vínculos a hojas de estilo CSS y documentos JavaScript, y metainformación dirigida al navegador web y a los *robots* de búsqueda (programas que rastrean las páginas web para indexar el contenido de las mismas en su buscador, por ejemplo Google). Las plantillas incluirán parte de esa información a través un archivo externo (mediante la función de PHP *include()*).

En el cuerpo del documento (<body>) se incluirá todo el contenido que se mostrará en la ventana del navegador web.

Una manera de poder controlar el aspecto visual es dotar al documento de una estructura de capas (mediante el uso de cajas, con etiqueta), y definir en CSS el aspecto y la ubicación de las mismas. De esta forma, si posteriormente se quiere añadir un elemento nuevo a la página web, tan sólo hay que introducir una nueva capa en la plantilla y definir sus características de presentación en la hoja de estilos CSS. Hay varias plantillas, que se usarán:

- Para página de presentación: dada la simplicidad deseada para esta parte, el cuerpo (<body>) no tendrá ninguna capa, y en él se cargará el módulo correspondiente a la presentación.

• Para el resto de páginas sin mapas: el cuerpo estará compuesto de las siguientes capas:

Figura 12.2. Capas de la plantilla general de la web.

La capa "main" contendrá el módulo cargado, mientras que las capas inferiores se encargarán de ofrecer distintos elementos visuales.

• Para el resto de páginas con un mapa interactivo: tiene una estructura similar a la anterior, con la de diferencia de que incluye una llamada a la función JavaScript que carga el mapa interactivo.

12.1.4 Organización en directorios

El diseño modular de la aplicación implicará separarla en varios archivos, que se organizarán en varias carpetas o directorios según su función, para facilitar su posterior manejo.

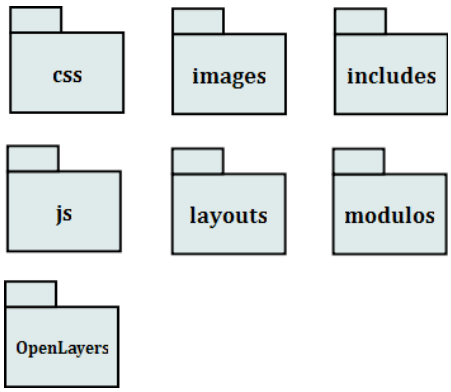

Figura 12.3. Organización en directorios de la aplicación web.

A continuación se describe el contenido de cada directorio:

- css/ : incluye las hojas de estilo CSS de la aplicación web.
- images/ : incluye todas las imágenes utilizadas por la aplicación web.
- includes/ : incluye los archivos PHP que se incluye en la estructura de la página, como la cabecera, el menú del usuario, o el pie de página.
- js/ : incluye todos los archivos JavaScript utilizados por la aplicación web.
- layouts/ : incluye las plantillas web, y también las distintas cabeceras XHTML (etiqueta <head>).
- modulos/ : incluye los módulos de la web. En apartados posteriores se describirán cada uno de ellos.
- OpenLayers/ : incluye la API OpenLayers.
- / (directorio raíz) : incluye los principales archivos (*index.php* y *conf.php*), así como los archivos utilizados por el servidor para gestionar sesiones de usuario y todo lo relacionado con la gestión de rutas.

12.2 Modelo de datos

En el modelo de datos se explicarán el modelo relacional de la base de datos y el modelo de clases implementadas en la aplicación.

12.2.1 Modelo relacional de la base de datos

El modelo relacional de una base de datos es el más actualizado actualmente. Fue creado por Edgar Frank Codd en la década de los setenta, y se trata un paradigma basado en la teoría de conjuntos y lógica de predicados de primer orden que proporciona una forma sencilla de representar los datos: mediante tablas bidimensionales, llamadas relaciones.

Como se ha visto en apartados anteriores, la aplicación web hace uso de la base de datos PostgreSQL (con la extensión PostGIS) tanto para la manipulación de datos espaciales como para la gestión de usuarios. El esquema de la base de datos es el siguiente:

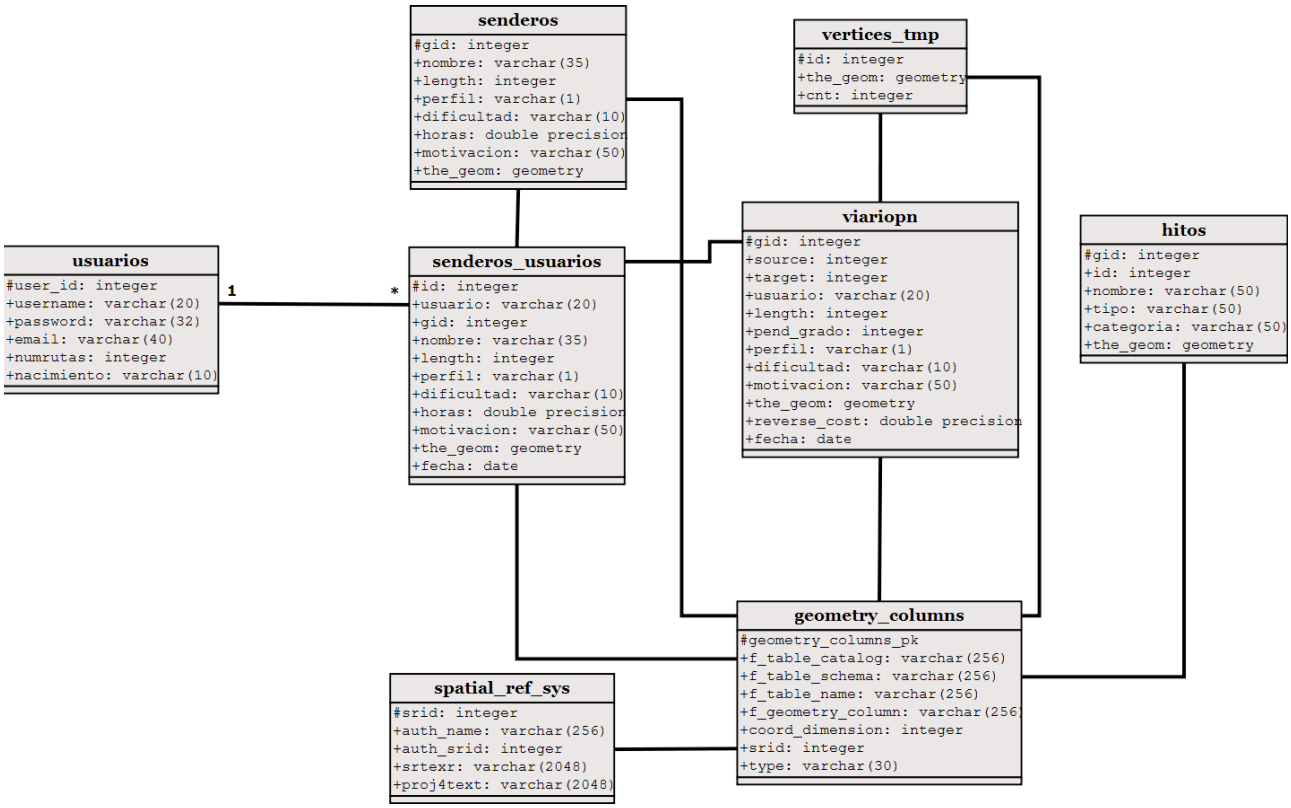

Figura 12.4. Esquema de la base de datos.

El papel de cada una de las tablas se resume en los siguientes puntos:

- La tabla *usuarios* guarda la información de los usuarios registrados en el sistema.

- La tabla *senderos_usuarios* almacena las rutas guardadas por los usuarios. La relación definida con la tabla *usuarios* establece que cada registros de *senderos_usuarios* pertenecerá a un sólo usuario, mientras que éste puede tener cualquier número (o cero) de registros en *senderos_usuarios*.

- La tabla *senderos* guarda las rutas senderistas más típicas y tradicionales de la zona. Se trata por tanto de rutas prefijadas. Cada una de ellas almacena una serie de atributos que definen sus

características, necesarias a la hora de realizar el cálculo de rutas recomendadas (motivación, horas, perfil, etc.).

• La tabla *viariopn* guarda todo el viario de rutas recopilado de la Sierra de las Nieves, y sobre estos datos se realizarán los cálculos de rutas recomendadas y ruta más corta.

• La tabla *hitos* guarda los puntos de interés de la Sierra de las Nieves, con objeto de aportar información adicional al usuario.

• La tabla *vertices_tmp* se guarda información relativa al cálculo de la topología del viario (realizado por pgRouting), en concreto los nodos del grafo generados y su posición geográfica. Se trata por tanto de una tabla temporal que no interviene en el funcionamiento de la aplicación, pero que puede resultar de utilidad al desarrollador, como se explica en el ANEXO I.

• Las tablas *spatial_ref_sys* y *geometry_columns* se generan al crear la base de datos en PostGIS, y guardan información relativa a los sistema de referencia espaciales y al tipo de datos geométricos de cada una de las tablas de la base de datos que incluya información geográfica (atributo "the_geom").

12.2.2 Modelo de clases

El modelo de clases se aplica a los scripts escrito en JavaScript. En realidad este lenguaje no maneja clases de una forma estricta (como Java), aunque implementa un modelo parecido, que a veces recibe el nombre de modelo de pseudoclases. En cualquier caso, se van a mostrar los métodos y las clases (o pseudoclases) más importante de la aplicación.

El modelo de clases de la aplicación se apoya en la librería OpenLayers, que contiene las clases que se muestran a continuación en el siguiente diagrama:

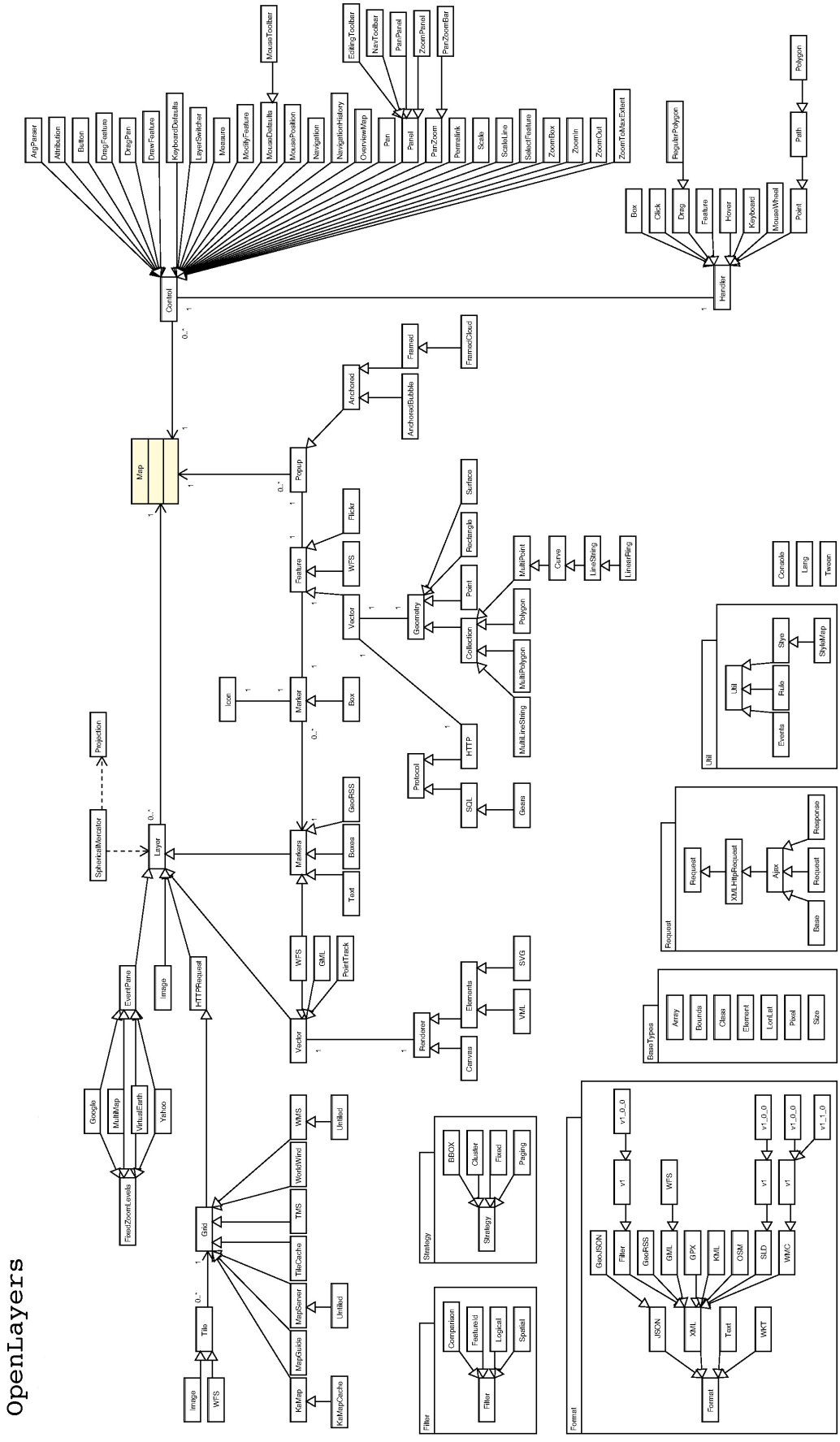

Figura 12.5. Diagrama de clases de OpenLayers.

Las clases de OpenLayers más utilizadas serán:

- *OpenLayers.Map*: para la definición del mapa.
- *OpenLayers.Layer*: para las capas del mapa.
- *OpenLayers.Projection*: para la proyección de los mapas.
- *OpenLayers.Control*: para los controles del mapa.
- *OpenLayers.Bounds*: para definir las coordenadas que delimiten la visualización del mapa.

12.3 Interacción entre la aplicación web y la IDE

En este apartado se abordará cómo interactúa la aplicación web con la IDE, destacando aspectos como la comunicación entre el navegador web y el servidor, o las operaciones relacionadas con rutas.

12.3.1 Mapas interactivos con OpenLayers

OpenLayers será el encargado de generar y hacer visibles los mapas interactivos con las rutas. Esta aplicación se ejecutará en el navegador del usuario mediante código escrito en JavaScript, y se le añadirán una serie de controles para poder navegar por el mapa, realizar zoom sobre el mismo, etc.

El primer paso será crear una instancia de mapa (mediante el constructor *new OpenLayers.Map*), y a continuación se añaden las capas base (ortofotos, topográfico, etc.). Para esto es necesario que OpenLayers se comunique vía WMS con la IDE (mediante el constructor *new OpenLayers.Layer.WMS*) y haga de ventana por la que el servidor de mapas (UMN MapServer) responda a la petición mostrando la capa y la zona deseadas.

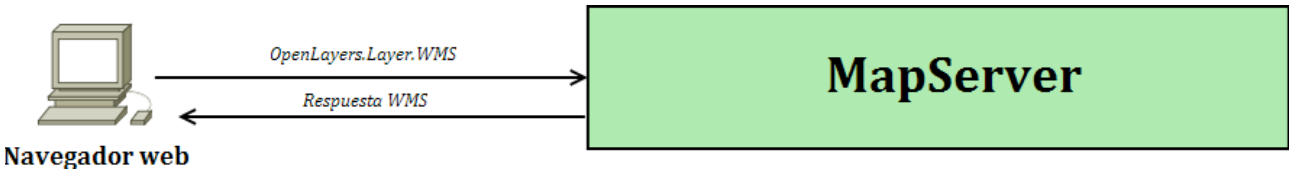

Figura 12.6. Comunicación entre OpenLayers y UMN MapServer.

Existe otro tipo de capas, que no van a ser generadas por MapServer, sino por el propio OpenLayers (y por tanto el procesamiento va a recaer en el navegador web): las rutas obtenidas como resultado de una búsqueda, o las rutas guardadas por el usuario en su lista personal. Para ello también necesita comunicarse con la IDE, y lo hará a través de una petición asíncrona *XMLHTTPRequest* (por medio de AJAX) a un módulo PHP del servidor que se encargará de dialogar con la base de datos PostGIS, y obtener los datos de las rutas. Este módulo responderá a OpenLayers a través de un objeto GeoJSON con todos datos referenciados de las rutas requeridas. OpenLayers se encargará de almacenar, manipular y dibujar la ruta en el mapa (mediante el constructor *new OpenLayers.Layer.Vector*). En el ANEXO II se adjunta una parte del código que ilustra todo este proceso.

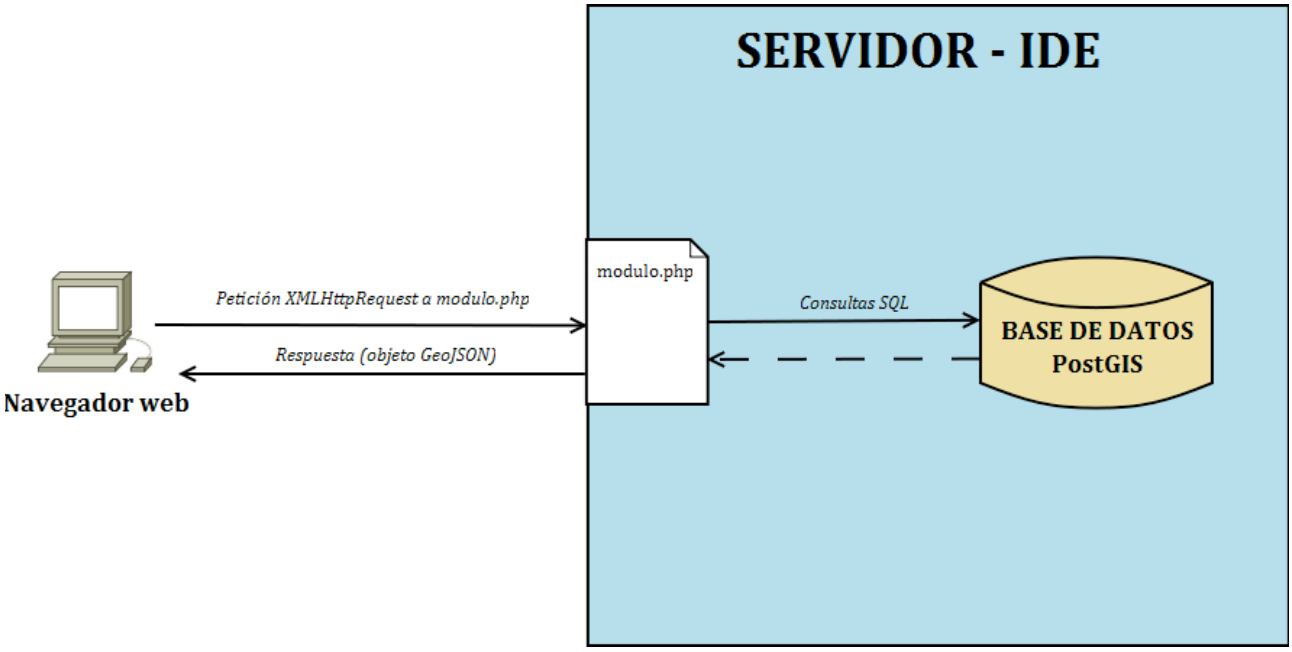

Figura 12.7. Comunicación entre OpenLayers y la IDE.

12.3.2 Tabla con rutas

La tabla con las rutas del usuario o las rutas resultado se generará dinámicamente, mediante el uso del interfaz DOM (Document Object Model), que proporciona una serie de funciones en JavaScript para acceder e interactuar con cualquier objeto del documento XHTML.

Como se acaba de comentar, el cliente (navegador), por medio de OpenLayers, realiza una petición mediante AJAX al servidor y recibe como respuesta datos en formato GeoJSON que el visor se encarga de manipular y transformar visualmente en rutas.

Paralelamente, se va generando una tabla a partir del objeto GeoJSON recibido, que incluye datos relativos a la longitud de la ruta, el duración estimada del recorrido de la misma, o la motivación principal de la misma. Y se añaden una serie de opciones, en caso de que el usuario esté autenticado, como bajar la ruta, agregarla a la lista personal, o borrarla. Cada fila se crea con la función del DOM *insertRow()*, y cada campo se va insertando en una celda. Para iluestrar esto último, el siguiente ejemplo de código inserta una celda en la fila (variable *nuevaFila*) con la información de la longitud de la ruta:

```
//insertamos celda en la posición 2 de la fila
var celda = nuevaFila.insertCell(2);

//añadimos un texto con la longitud de la ruta i (dato almacenado
//en el objeto GeoJSON 'JSONrutas'
celda.appendChild(document.createTextNode(JSONrutas.ruta[i].longitud));
```

Este tipo de comunicación asíncrona permite realizar cambios en la página web mostrada sin necesidad de recargarla por completo.

12.3.3 Cálculo de rutas recomendadas

La aplicación tiene prevista la incorporación de un algoritmo que calcula las rutas recomendadas, en función de las características suministradas por el usuario: perfil, motivación o motivaciones, y opcionalmente el tiempo disponible para realizar la ruta. Además, incorporará el cálculo de rutas típicas prefijadas que se ajusten a esas características.

La comunicación con la IDE se hace nuevamente en OpenLayers por medio de AJAX, a través de una petición *XMLHTTPRequest*, a un módulo PHP del servidor que recogerá los parámetros que le pasa OpenLayers (el perfil, las motivaciones y el tiempo disponible) y realizará los cálculos necesarios, apoyándose en la base de datos PostGIS, para obtener las rutas que mejor se ajusten al usuario.

Este módulo, igualmente, responderá a OpenLayers a través de un objeto GeoJSON con todos datos geoespaciales y otras características de las rutas.

12.3.4 Cálculo de la ruta más corta

El modo de comunicarse con la IDE es similar al apartado anterior. Cuando el usuario señala en el mapa los puntos de origen y destino y realiza la petición de cálculo de la ruta, OpenLayers se comunica mediante AJAX con un módulo que conecta con PostGIS, y éste realice los cálculos con las funciones suministradas por pgRouting.

La respuesta, nuevamente, se produce en forma de objeto GeoJSON, que OpenLayers se encargará de pintar en el mapa.

12.3.5 Descargar, agregar o borrar ruta

Cuando un usuario pulsa el botón de descarga la ruta, el evento que escucha y controla a ese botón reacciona y se ejecuta una función asociada a ese evento. Dicha función llamará a un módulo PHP (mediante AJAX) para que dialogue con PostGIS y éste convierta los datos de la ruta a formato KML.

Agregar una ruta funciona exactamente igual que descargar una ruta, sólo que los datos de la ruta se guardan en la lista del usuario en la base de datos. Y lo mismo ocurre a la hora de borrar una ruta, con el resultado de que PostGIS elimina la ruta de la lista personal de rutas del usuario que está almacenada en la base de datos.

12.4 Módulos de la aplicación

12.4.1 Página de presentación

La página de presentación es más una presentación visual de la web. Contiene un menú con los enlaces a la página principal, búsqueda de rutas, información sobre la Sierra de las Nieves, y un enlace "acerca de" que al pulsar sobre él muestra una ventana emergente explicando brevemente en qué consiste la web. El fondo

contiene una imagen de la Sierra de las Nieves, que se reescala según el tamaño de la ventana del navegador (por medio de CSS).

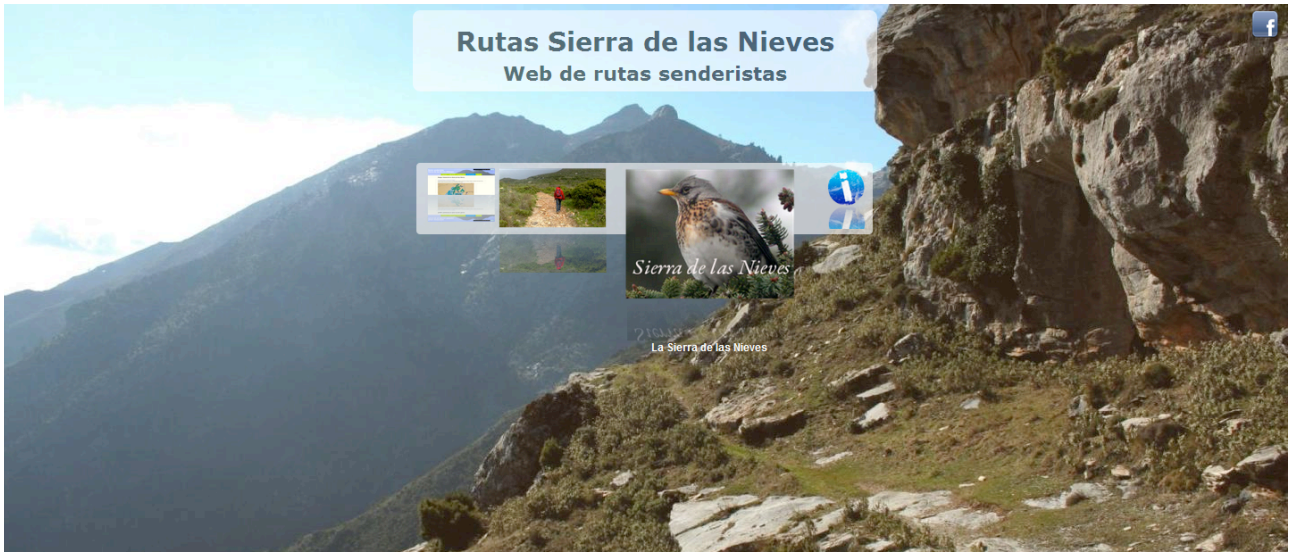

Figura 12.8. Página de presentación.

12.4.2 Página principal

Aquí se explica más a fondo, en un recuadro con pestañas (programado con las librería jQuery) las características de la aplicación web, en qué consiste el buscador de rutas y a modo de tutorial se indica cómo usar las diferentes herramientas que ofrece, con objeto de que el usuario se familiarice lo antes posible con la web. También se invita al usuario a registrarse.

Figura 12.9. Página principal.

12.4.3 Buscador de rutas

Dado que en la página principal se explica con todo detalle qué ofrece el buscador, aquí se incluyen dos botones para acceder a los dos tipos de búsqueda, con un pequeño mensaje de información (realizado con jQuery) que explica brevemente su significado cuando se pasa el cursor por encima del botón correspondiente.

Figura 12.10. Buscador de rutas.

12.4.4 Rutas recomendadas

En esta página se incluirá un formulario dinámico en función del tipo de usuario que acceda a ella:

- Usuario anónimo: se pide introducir un perfil, motivaciones, y opcionalmente el tiempo disponible y la fecha prevista para recorrer la ruta.

- Usuario autenticado: se muestra el perfil actual del usuario, y se ofrece la posibilidad de introducir un perfil temporal. Se pide introducir motivaciones, y opcionalmente el tiempo disponible y la fecha prevista para recorrer la ruta.

Este formulario implementa varias funciones propias en JavaScript (para que las motivaciones "Deporte" y "Paseo" sean excluyentes), y un validador (jQuery Validation, bajo licencia GPL), que controlará que el usuario haya introducido un perfil y al menos una motivación, y que el dato introducido en el tiempo disponible sea un número.

Una vez rellenado el formulario y enviado, la aplicación nos llevará a otra página mostrando las rutas recomendadas calculadas.

Figura 12.11. Página de rutas recomendadas.

12.4.5 Rutas recomendadas calculadas

En esta página se mostrará el mapa con las rutas resultado listadas en una tabla, en la cual se especifica la longitud, tiempo estimado y otros parámetros de cada una de las rutas. Si el usuario está autenticado, tendrá la posibilidad de descargar o agregar la ruta a su lista.

Además, si el usuario introdujo la fecha que tenía previsto realizar la ruta, la aplicación le informará de las horas en que amanezca y atardezca ese día, así como las horas de sol.

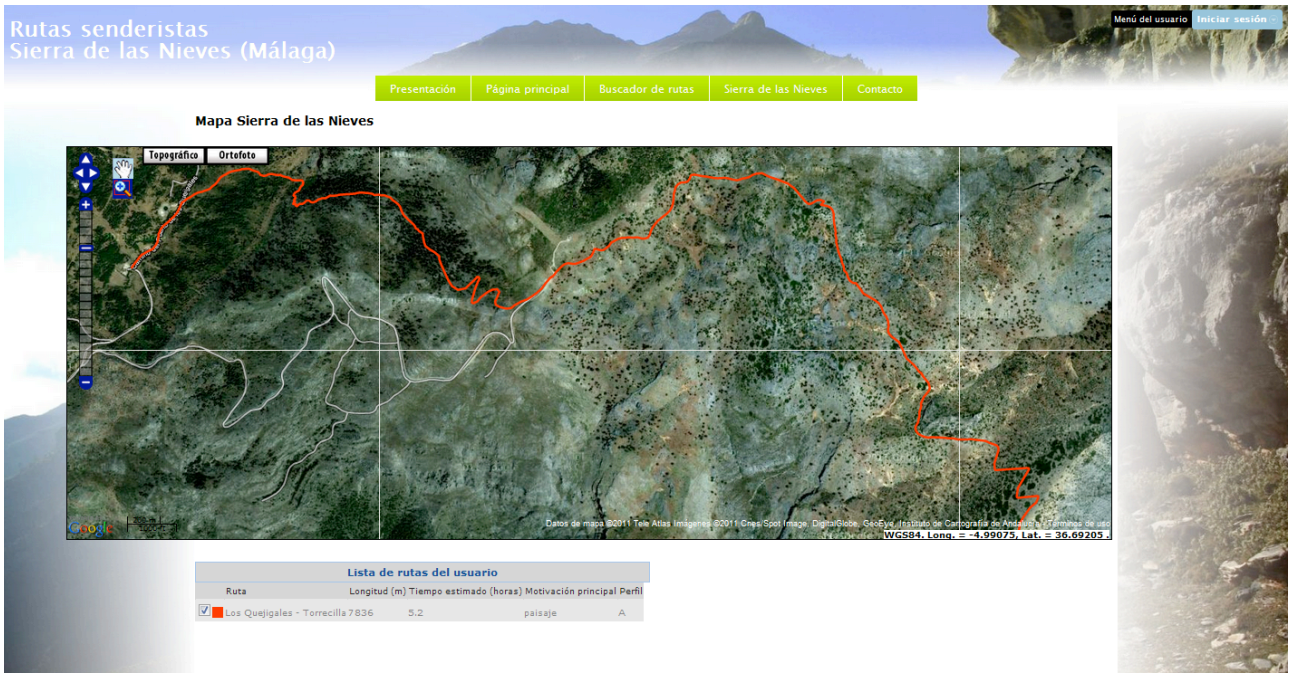

Figura 12.12. Página de resultados de rutas recomendadas.

12.4.6 Ruta más corta

Esta página mostrará un mapa con el viario completo de rutas de la Sierra de las Nieves, y se ofrecerá al usuario indicar los puntos de origen y destino en el mapa para calcular la ruta más corta entre esos puntos (siempre que ambos estén interconectados).

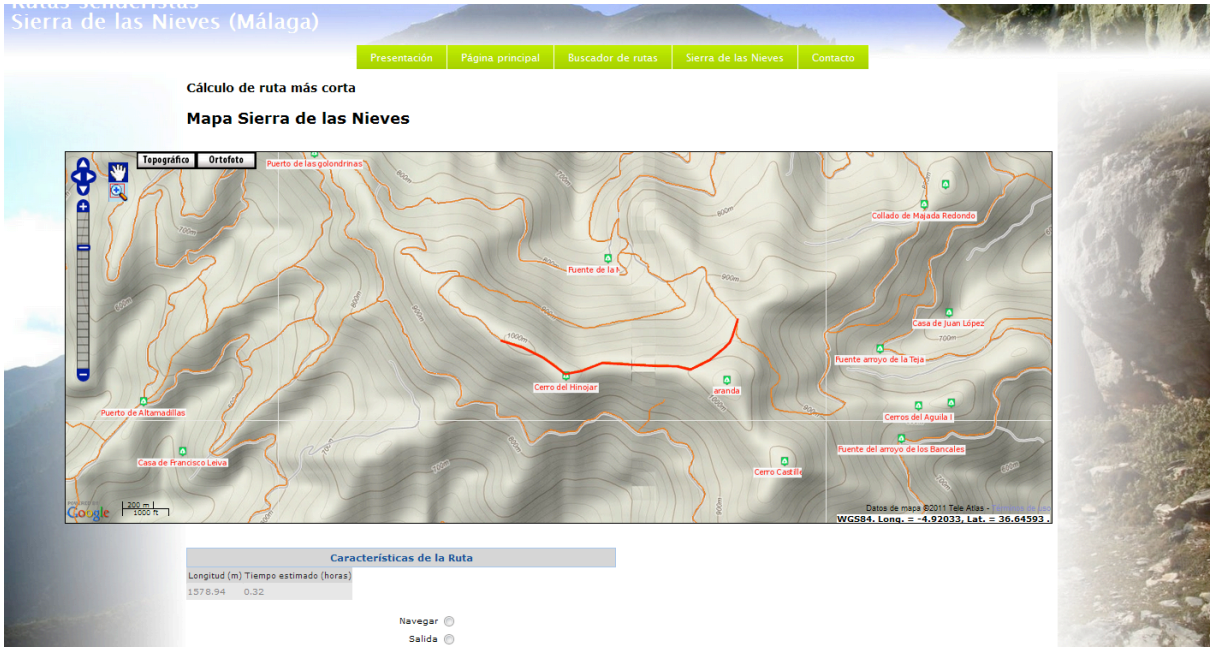

Figura 12.13. Página de ruta más corta.

.

12.4.7 Página de información de la Sierra de las Nieves

Es importante dar información sobre el entorno donde transcurren las rutas, y esta página contiene información general sobre el Parque Natural de la Sierra de las Nieves. Al final de la misma se incluyen varios enlaces relacionados con este parque natural.

Figura 12.14. Página de información de la Sierra de las Nieves.

12.4.8 Panel de Control

En esta página, sólo accesible por un usuario autenticado, se ofrecen las principales opciones de configuración de todo lo relacionado con la cuenta del usuario:

- Gestión de rutas.
- Editar perfil.
- Editar datos personales.

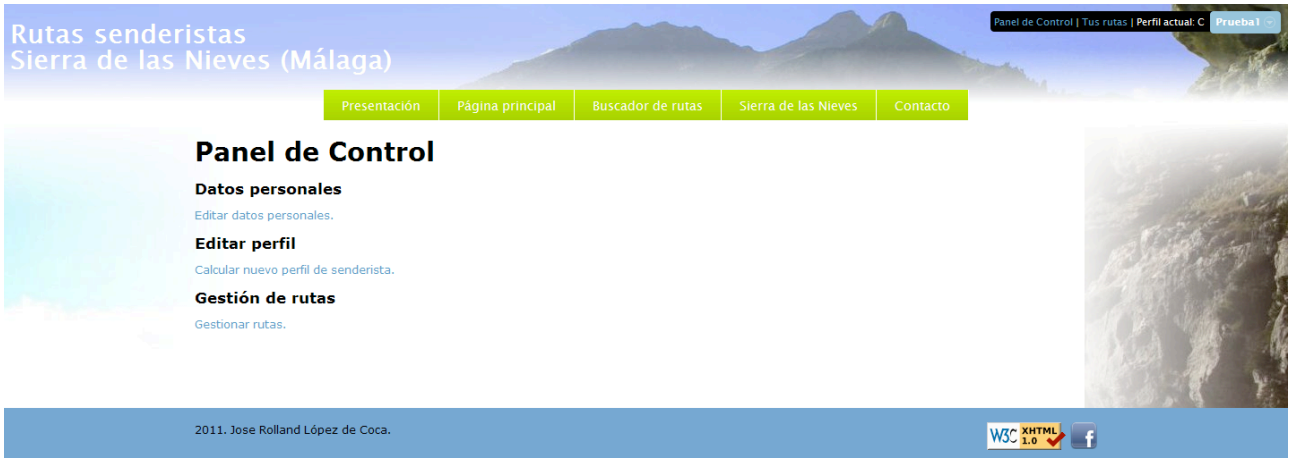

Figura 12.15. Página de panel de control.

12.4.9 Gestión de rutas

En este módulo, el usuario podrá consultar en un mapa interactivo las rutas que ha guardado en su lista personal, y podrá descargarlas en KML o borrarlas de la lista.

Figura 12.16. Página de gestión de rutas.

12.4.10 Editar perfil

En esta página habrá un formulario para calcular de nuevo del perfil del usuario.

Figura 12.17. Página de edición de perfil.

12.4.11 Editar datos personales

En esta página habrá un formulario para editar los datos personales (contraseña y correo electrónico) del usuario.

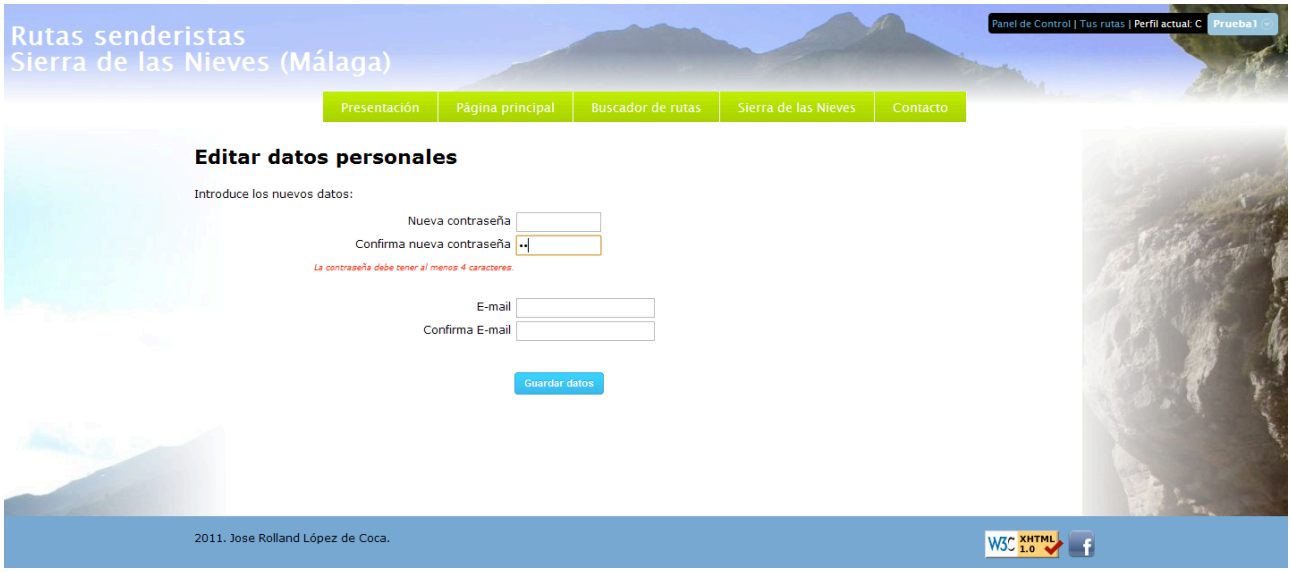

Figura 12.18. Página de edición de datos personales.

12.4.12 Página de contacto

En esta página se ofrece el correo electrónico del administrador de la página, así como un resumen de qué ofrece la web.

Figura 12.19. Página de contacto.

12.4.13 Página de registro

El módulo de registro muestra un formulario, que también hace uso del validador jQuery Validation, donde se piden una serie de datos para proceder al registro del usuario en el sistema.

Figura 12.20. Página de registro.

12.4.14 Panel de administrador

El panel del administrador permite consultar todos los usuarios que están registrados en la web, y borrar cualquiera de ellos (menos al propio administrador).

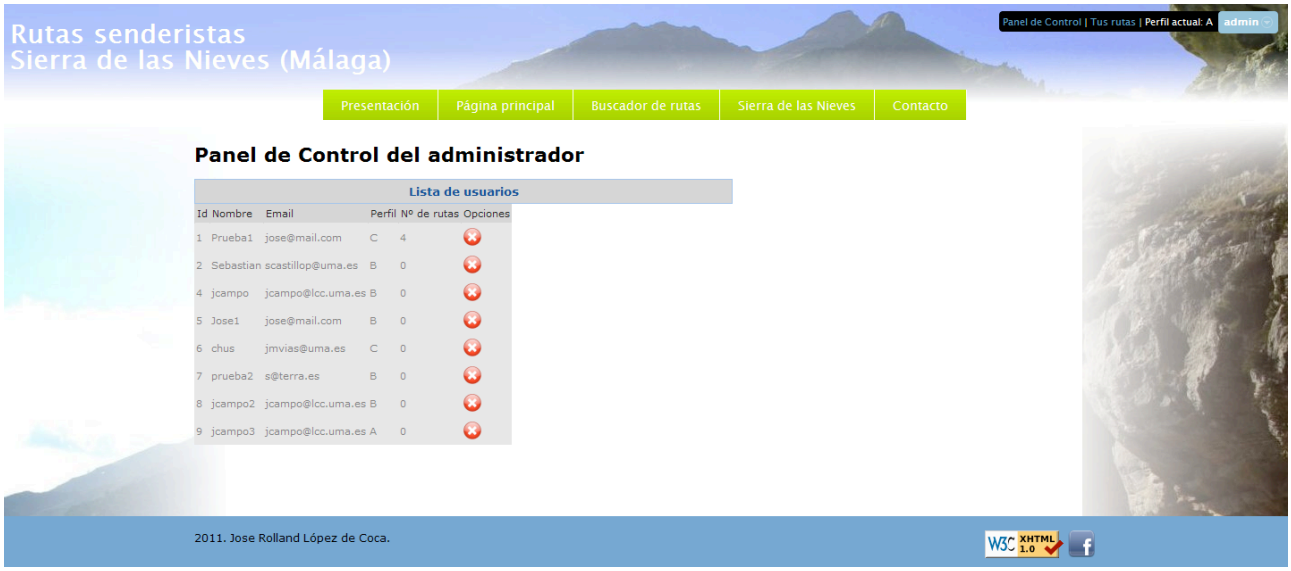

Figura 12.21. Página de panel del administrador.

12.4 Gestión y control de usuarios

La aplicación ofrece la posibilidad de gestionar una comunidad de usuarios. Cada usuario registrado tendrá acceso a nuevas funcionalidades de la aplicación, como pueden ser el cálculo de su perfil, el guardado de rutas o la descarga de las mismas.

Para ello, se creará una tabla concreta (*usuarios*) en la base de datos PostgreSQL que guardará en cada registro todos los datos del usuario, como su nombre, su contraseña, o su correo electrónico.

Las contraseñas de los usuarios serán guardadas en la base de datos mediante una *huella* digital, usando el algoritmo MD5. Básicamente, MD5 es una función hash unidireccional. De esta forma, cuando el usuario inicie sesión enviando su usuario y contraseña, el sistema calculará el código MD5 de esa contraseña y comprobará si coincide con el código guardado en la base de datos. Así, nadie, incluido el administrador, podrá saber las contraseñas mirando en dicha base de datos.

Por otra parte, las rutas guardadas por cada usuario se guardarán en otra tabla (*senderos_usuarios*) con toda la información geoespacial de las rutas, así, como información adicional como la longitud, la dificultad, el perfil, o el tiempo estimado.

La aplicación gestionará distintos perfiles de usuario, controlando el acceso de los usuarios a los distintos módulos. Como se ha comentado anteriormente, los archivos principal (*index.php*) y de configuración (*conf.php*) de la aplicación serán los encargados de ello.

Cuando un usuario inicia sesión en la aplicación, el servidor crea una serie de variables de sesión de PHP vinculadas a ese usuario, y cuando cierra sesión, el servidor las destruye. Para controlar que un usuario ha iniciado sesión, el servidor, mediante PHP, comprobará que existen las variables de sesión correspondientes, de la siguiente forma:

- El primer paso es iniciando una sesión en PHP:

```
session_start();
```

- Después comprueba que la variable de sesión del usuario *$_SESSION['SESSION_UNAME']* está creada:

```
if (isset($_SESSION['SESSION_UNAME']))
{
     //el usuario está autenticado
}
else
{
     //el usuario no está autenticado
}
```

En caso de que un usuario intente acceder a un módulo en el que no tenga permiso para ello, se mostrará una nueva página con un mensaje informando de la imposibilidad de entrar a dicho módulo.

CAPÍTULO

13 Diseño del interfaz de usuario

Tras haber realizado una primera aproximación en el capítulo 10 analizando cómo iba a ser la interfaz, en el presente capítulo se describirá el aspecto final de la interfaz de usuario.

13.1 Diseño del interfaz

El diseño del interfaz va a ser desarrollado mediante las hojas de estilo CSS y la librería jQuery (escrita en JavaScript, y compatible con la gran mayoría de navegadores web).

La hoja de estilos CSS es la herramienta fundamental para definir dónde y de qué forma se ubicarán las capas incluidas en las plantillas así como cualquier tipo de objeto del documento, y también especificará el tipo de letra, el tamaño, los márgenes, los párrafos, los botones, etc. En definitiva, especificará la presentación visual de la web en los aspectos más básicos.

La librería jQuery complementa el trabajo de CSS con operaciones más complejas que por el momento no es capaz de hacer CSS, como por ejemplo la transición animada de color cuando se pasa el cursor del ratón por encima de un botón del menú principal de navegación. Se trata por tanto de un aditivo que aporta un mayor atractivo visual y dinamismo a la navegación.

Los dos formatos de imagen escogidos para los gráficos e imágenes incluidas en la web son JPEG (Joint Photographic Experts Group) y PNG (Portable Networks Graphics), muy extendidos en la Web. El primero sirve para guardar imágenes de gran tamaño (como los fondos) ocupando muy poco espacio (debido a su gran capacidad de compresión), a costa de una ligera pérdida de calidad. El segundo no tiene tal capacidad de compresión, pero es útil para mostrar pequeños gráficos sin pérdida alguna de calidad y además soporta el uso de transparencias.

Para la creación y edición de gráficos, imágenes, iconos y botones del interfaz se va a hacer uso del editor de imágenes Gimp 2. Las principales operaciones realizadas son redimensionar imágenes, corregir los niveles de color, crear los botones del interfaz, y aplicar reflejos y degradados.

A continuación se describen los principales elementos del interfaz de usuario de la aplicación web.

13.1.1 Menú de la página de presentación

El principal elemento de la página de presentación es su menú, cuyo diseño está apoyado en unas funciones de jQuery (Interface Elements for jQuery, bajo licencia GPL), que mediante una animación aumenta el tamaño del icono por el que pasa el cursor. La motivación de este diseño se basa en intentar captar la atención del usuario, con enlaces a las principales funcionalidades de la aplicación web.

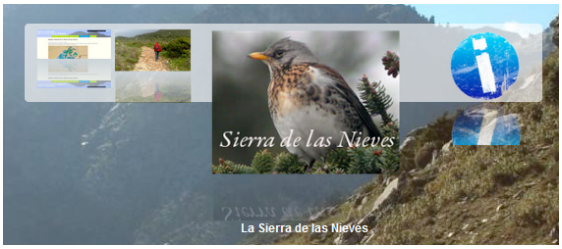

Figura 13.1. Menú de la página de presentación.

13.1.2 Menú de usuario

El menú de usuario acompaña durante toda la navegación por la web. Se trata de un menú desplegable que se encuentra en la parte superior derecha de la pantalla, y en él se puede iniciar o finalizar sesión, o acceder directamente a las páginas personales del usuario, tanto en el menú desplegable como a la izquierda del menú:

- Acceso directo al panel de control del usuario ("Panel de Control").
- Acceso directo a la lista de rutas del usuario ("Tus rutas").

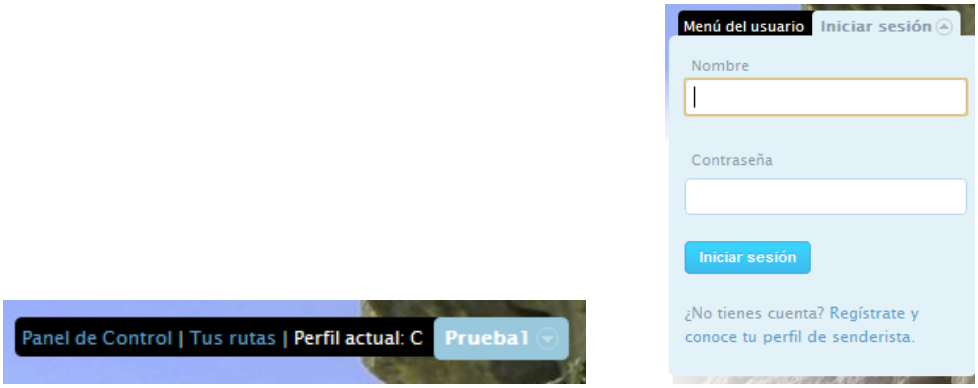

Figura 13.2. Menú de usuario.

También incluye un enlace para registrarse en la web, si el usuario es anónimo.

13.1.3 Menú de navegación

El menú de navegación también se encuentra en todos los módulos de la aplicación web, a excepción de la presentación. Su objetivo es ofrecer un menú con los enlaces a los principales módulos de la aplicación:

- Presentación.
- Página principal.
- Buscador de rutas.
- Sierra de las Nieves.
- Contacto.

El diseño de este menú incluye una animación que hace cambiar de color al botón seleccionado por el cursor, y también incluye un código escrito en JavaScript para cambiar el estado de un botón (en concreto añade una clase al botón, "activo", que en CSS define un color distinto que el resto de botones) cuando está en la página a la que apunta ese botón:

```
$(document).ready(function() {
    jQuery
     ('#menua a[href$="'+window.location.search+'"]')
     .parent()
     .addClass("activo");
});
```

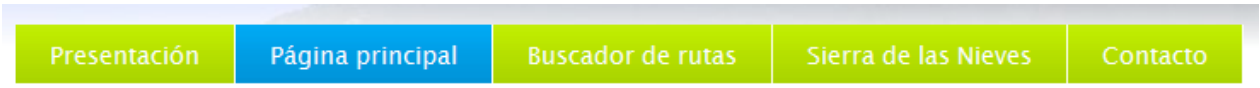

Figura 13.3. Menú de navegación (visto desde el módulo de la página principal).

13.1.4 Controles del mapa

Los controles del mapa permiten interactuar con él y aportar información adicional a lo que se está visualizando. La mayoría están incluidos en OpenLayers (como la barra de zoom), aunque esta librería suministra herramientas para personalizar sus controles o crear otros nuevos. Estos son los controles incluídos:

- Control de navegación: en realidad incluye dos tipos de navegación, usando los controles (las cuatro flechas situadas en la parte superior izquierda del mapa) o arrastrando el mapa con el ratón.
- Barra de zoom: permite realizar distintos grados de zoom sobre el centro del mapa.
- Caja de zoom: está situado en la parte superior izquierda (una lupa sobre un cuadrado), y permite realizar un zoom sobre el recuadro que dibuje el usuario con el ratón. La forma de dibujar el recuadro consiste en pulsar el botón principal del ratón sobre un punto del mapa, y moverlo hacia otro punto, trazando la diagonal del rectángulo sobre el que se realizará el zoom.
- Barra de escala: indica la escala a la que está el mapa mostrado (se indica en kilómetros y en millas).

• Coordenadas: indican la longitud y latitud geográficas del punto sobre el que está situado el cursor, de acuerdo al datum WGS84. Este control ha sido editado para mostrar claramente la información.

• Cambio de capa base: situado en la parte superior izquierda, permite cambiar la capa base entre un mapa topográfico y un mapa con fotografías tomadas desde al aire (ortofoto). Se trata de un control creado desde cero, que incluye unos botones con el nombre de la capa.

Figura 13.4. Controles del mapa.

13.1.5 Tabla de rutas

La tabla de rutas ofrece un listado con información de las mismas (nombre, longitud, etc.), y una serie de controles para mostrarlas en el mapa, descargarlas, borrarlas o agregarlas a la lista personal. Además, incluye una función de jQuery llamada Tablasorter (bajo licencia GPL) que permite ordenar la tabla de acuerdo a cualquier campo, pulsando sobre la cabecera de la columna que se desee ordenar.

Ruta	Longitud (m)	Tiempo estimado (horas)	Motivación principal	Perfil	Fecha	Opciones
Caucón - Tajo de la Caina	4409	2.9	paisaje	C	2011-05-30	
La Fuenfría	3376	2.3	paseo	C	2011-05-30	
Puerto Saucillo - Puerto Bellina	3239	2.2	paseo	C	2011-05-30	
La Rejía	4444	3	ecologico	C	2011-06-22	

Figura 13.5. Tabla de rutas.

13.1.6 Mensajes de información

Algunos botones y formularios incluyen pequeños mensajes de información para explicar qué hacen o en qué consisten algunos campos de esos formularios. Los mensajes están programados con unas funciones de la librería jQuery (Tiptip, bajo licencia GPL).

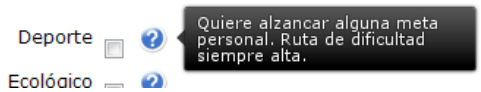

Figura 13.6. Mensaje de información.

13.1.7 Fechas

Cuando la aplicación pide introducir una fecha, incluye una pequeña utilidad programada con la librería jQuery UI (una versión de jQuery especializada en interfaces de usuario, bajo licencia GPL) llamada Datepicker, que muestra un calendario gráfico donde se selecciona la fecha pulsando sobre el día del mes y año deseados. Esto evita por ejemplo que el usuario pueda introducir una fecha con un formato erróneo al esperado por la aplicación.

Figura 13.7. Cuadro de calendario para seleccionar fecha.

13.2 Usabilidad del interfaz

En el capítulo 7 se definió el concepto de usabilidad, y también se citaron una serie de características contempladas para mejorar la usabilidad de un interfaz. En este punto se van a enumerar las medidas adoptadas para cumplir los requisitos de usabilidad marcados.

El objetivo principal de la usabilidad se centra en la rapidez y facilidad de uso de la aplicación. Durante el desarrollo del interfaz se han tenido presentes estos dos puntos:

- Rapidez: se ha evitado el uso de imágenes de gran tamaño o en gran número en una misma página, así como de *scripts* no programados en JavaScript (como *applets* de Java, videos, o animaciones Adobe Flash) que puedan ralentizar la carga y la navegación por las páginas, optando por un diseño lo más minimalista posible.

- Facilidad de uso: en la página principal se explica brevemente cómo usar la aplicación con objeto de que el usuario comience a usarla correctamente lo antes posible (l*earnability*), y se han intentado implementar menús intuitivos y claros, con accesos directos adicionales que ayuden a una navegación sin dificultades y fácil de aprender por parte del usuario (*memorability*).

Otros puntos que había que tener en cuenta son:

- Consistencia: la aplicación muestra un diseño consistente, ya que todos los módulos tienen el mismo diseño, salvo la página de presentación, que realmente no participa en las principales funciones de la aplicación.

- Reversibilidad: la aplicación web permite retroceder en el historial de navegación en la mayoría de los casos.

- Ausencia de errores: gran parte de las pruebas de la aplicación han ido dirigidas a comprobar el correcto funcionamiento del interfaz, y la ausencia de errores que entorpezcan la navegación.

- Composición de la página (medidas cualitativas): en cada página se ha optado por un diseño minimalista. Esto implica un tamaño limitado de la página, evitando en la medida de lo posible hacer uso del *scroll* (deslizamiento) vertical, con el mínimo texto necesario, en definitiva, evitar que esté excesivamente recargada y penalice la facilidad de uso y aprendizaje del usuario.

- Formato de la página (medidas cualitativas): la elección de la paleta de colores es importante para el aspecto visual de la web. En este caso se ha evitado el uso de colores excesivamente llamativos, y se han usado degradados y transparencias para suavizar los contrastes. También se ha elegido un tipo de letra estándar lo más sencilla posible, sin bordeados.

BLOQUE 4: PRUEBAS

CAPÍTULO

14 Pruebas unitarias

Las pruebas unitarias, también llamadas de caja blanca o modulares (ya que nos permiten determinar si un módulo de la aplicación está listo y correctamente terminado), son las primera etapas de pruebas a considerar en el desarrollo de una aplicación informática. Estas pruebas se centran probar toda la funcionalidad del módulo puesto a prueba, y para ello es recomendable que dicho módulo no sea excesivamente complejo y su depuración se complique en demasía. Si el desarrollador se encuentra con un módulo complejo, puede separarlo en varios módulos y así facilitará la posterior fase de pruebas.

En definitiva, el objetivo final de estas pruebas unitarias es confirmar el correcto funcionamiento de las diferentes partes de la aplicación web, y también de la IDE que la integra.

Durante el desarrollo del proyecto se realizaron en primer lugar diferentes pruebas del sistema:

• Servidor web: una vez instalado el servidor web Apache y del módulo PHP para Apache, se comprobó su funcionamiento creando una página simple en HTML con un código en PHP que incluyera la función *phpinfo()* (que muestra información de configuración de Apache y PHP). Dicha página pasó a visualizarse en el navegador a través del servidor, y mostró la información esperada.

• Servidor de mapas: tras instalar MapServer, se comprobó su correcto funcionamiento realizando una petición a su módulo CGI a través del servidor (http://dominio/cgi-bin/mapserv). La respuesta dada fue la esperada, mostró el siguiente texto: "No query information to decode. QUERY_STRING is set, but empty.", que significa que hemos realizado una petición vacía.

• Base de datos: después de instalar PostgreSQL, PostGIS y pgRouting, se creó una base de datos con la plantilla de PostGis y las funciones de pgRouting y se comprobó que generaba las tablas y creaba las funciones esperadas.

• Datos geoespaciales: se realizó el volcado de los datos geoespaciales en la base de datos creada que iban a ser manipualdos por la misma, y se visualizaron con la aplicación SIG de escritorio gvSIG, para verificar que el proceso había tenido éxito. A continuación se creó un archivo de configuración

Mapfile y una página web con lo mínimo indispensable para mostrar los datos geoespaciales a través del servidor de mapas.

A la hora de implementar la aplicación web, se planificación el proceso de prueba de la siguiente forma:

- En primer lugar, probar la configuración general de los módulos, comprobar que la aplicación carga bien los módulos y las plantillas correspondientes.
- Comprobar que el contenido mostrado por cada módulo es el adecuado.
- Probar individualmente cada módulo de la aplicación, con todos los posibles perfiles de usuario (anónimo, autenticado, administrador), y comprobar permisos.
- Las pruebas se realizan de forma progresiva, probando cada función que se añada en el módulo.
- Análisis de los resultados de las pruebas, verificación y correcciones en caso necesario.

Tras realizar el primer paso, el segundo paso siempre consistía en un análisis visual de lo que mostraba el navegador en la pantalla. Después se hicieron las pruebas sobre cada módulo, entre las que destacan:

- Módulos de registro y rutas recomendadas: comprobación del validador del formulario, introduciendo distintos conjuntos de datos, tanto erróneos como válidos, y análisis de la respuesta dada. Probar la comunicación con la base de datos. En el caso del registro, registrando un usuario y verificando que se ha añadido en la tabla correspondiente de la base de datos. En el caso de las rutas recomendadas, comprobando en un script PHP que los datos son enviados de forma correcta.
- Módulo de resultado de rutas recomendadas: pruebas con distintas combinaciones de datos de entradas (procedentes del módulo de rutas recomendadas), y verificación. Pruebas individuales con el script de cálculo de rutas, y análisis de la comunicación entre el servidor y OpenLayers, y del objeto GeoJSON recibido. Se comprobó que el fichero KML descargado de una ruta funcionara en Google Earth, y mostrara los datos correctamente.
- Módulo de ruta más corta: pruebas en diferentes zonas del viario de rutas, y fuera del mismo, y análisis de la comunicación con la base de datos y la respuesta dada.
- Módulos de edición de perfil y datos personales: crear un usuario y realizar distintas pruebas, comprobando que los cambios en los datos se registran en la base de datos. También se comprobó el correcto funcionamiento del validador del formulario.
- Módulo de panel de control del administrador: comprobar que se muestran todos los usuarios registrados en la base de datos, comprobar la validez de los datos, y realizar pruebas de borrado de usuario.

Muchas de estas pruebas se hicieron utilizando la herramienta depuración para el navegador Mozilla Firefox llamada Firebug.

CAPÍTULO

15 Pruebas funcionales

Las pruebas funcionales o de requisitos funcionales del sistema van a probar y validar el software contra los requisitos definidos en la fase de especificación y análisis del proyecto, poniendo en práctica los casos de uso contemplados.

Se realizaron pruebas para todos los casos de uso, y se comprobó que el flujo seguido es similar al plasmado en los diagramas de secuencia de dichos casos de uso. Todos produjeron un resultado correcto.

Además, el cliente de la aplicación se involucró en esta fase de pruebas, utilizando y comprobando las funcionalidades de la aplicación.

BLOQUE 5: CONCLUSIONES

CAPÍTULO

16 Conclusiones

Tras la finalización de todas las fases de desarrollo del proyecto, se expondrán una serie de conclusiones relativas a los objetivos planteados, a las pruebas realizadas, y por último una valoración personal de todo el proceso.

16.1 Conclusiones sobre objetivos planteados

En los primeros capítulos de la presente memoria se han expuesto los pasos a seguir para la realización de este proyecto y los objetivos marcados. Tras haber finalizado el proyecto creo que esos objetivos se han cumplido con creces:

- Se ha realizado un estudio previo del ámbito de los Sistemas de Información Geográfica y las Infraestructuras de Datos Espaciales, se han estudiado algunos conceptos necesarios de Cartografía; en definitiva, he adquirido una base teórica imprescindible para llevar a cabo el proyecto.
- Se han estudiado las opciones existentes para llevar a cabo el diseño de un nodo IDE.
- Se han estudiado y analizado paradigmas de desarrollo web y los lenguajes de programación necesarios para poder afrontar el desarrollo de la aplicación.
- Se ha implementado un nodo IDE totalmente operativo.
- Se ha creado una aplicación web que ofrece búsquedas personalizadas de rutas en base a unos parámetros proporcionados por el usuario (sus capacidades físicas, su motivación, etc.).
- Se ha logrado implementar un sistema de gestión de usuarios en la aplicación.
- Se ha logrado que la aplicación muestre las rutas de forma que sean útiles al usuario, ofreciendo la posibilidad de descargarlas, o almacenarlas en una lista personal dentro de la web.

- Se ha dotado a la aplicación de un interfaz de usuario lo más amigable, atractivo y sencillo posible.
- Se ha logrado interconectar la aplicación web con el nodo IDE implementado.

16.2 Conclusiones de la fase de pruebas

La fase de pruebas se ha realizado paralelamente a la fase de implementación del nodo IDE y la aplicación, intentando abarcar los principales casos de uso y posibles ejecuciones de la aplicación, y se ha comprobado que la aplicación funciona correctamente, mostrándose muy estable en todo momento. Sin duda ha podido quedar alguna posible ejecución sin estudiar, y no se puede asegurar por completo que la aplicación esté libre de errores, pero se ha intentado en todo momento estudiar los casos más importantes con multitud de variantes, poniendo especial atención en todo lo relacionado con el cálculo y manejo de rutas.

También se ha puesto énfasis en la compatibilidad de la aplicación con los principales navegadores web, repitiendo pruebas en los distintos navegadores. En ocasiones algún navegador se ha comportado de manera distinta a lo esperado (principalmente procesando código JavaScript y CSS), pero se ha logrado subsanar con ayuda de una nueva función o definición.

16.3 Conclusiones personales del proyecto

Mis sensaciones tras finalizar este proyecto son muy positivas. Ha sido un largo y duro camino, pero creo que ha merecido la pena.

He tomado contacto y aprendido un ámbito de la Informática, el de los Sistemas de Información Geográfica, que desconocía por completo. La Geografía, la Cartografía y todo lo relacionado con el estudio de mapas son cuestiones que me han apasionado desde pequeño, y me alegra haber podido satisfacer mi curiosidad por estos temas y haberlos podido incluir en un proyecto de Informática.

Además, siempre tuve inquietud por saber cómo funcionaban por dentro aplicaciones como Google Maps. Tras la realización de este proyecto, al menos ya empiezo a intuir cómo lo hace, y me parece todavía más apasionante y admirable que antes.

Por otra, he aprendido desde cero varios lenguajes de programación actuales y paradigmas de desarrollo web, así como en la puesta a punto y manejo de un servidor web y toda la tecnología relacionada con los SIG y las IDE. También he podido experimentar de primera mano los problemas que surgen cuando cierto navegador no cumple los estándares web.

He de admitir que al principio no veía cómo iba a ser capaz de aprender de forma autodidacta todos esos lenguajes (como HTML, CSS, PHP, y especialmente JavaScript), me parecía una tarea enorme. Pero a medida que los he ido estudiando, he notado varias cosas que me han aportado el estudio de esta Ingeniería Técnica: pese a no ver visto ni una línea de código de esos lenguajes durante los tres años de carrera, he reconocido patrones, similitudes en la sintaxis y en la forma con otros lenguajes (PHP es parecido a C y C++, JavaScript usa algo parecido a los objetos y las clases de Java, etc.), que me han facilitado su aprendizaje y posterior desarrollo; o la capacidad de afrontar un problema, planteándome de qué forma lo programaría. Es decir, me han aportado una serie de herramientas básicas para afrontar nuevos paradigmas de desarrollo de software.

CAPÍTULO

17 Líneas futuras

Aunque se han cumplido todos los objetivos planteados al inicio, durante el desarrollo del proyecto han surgido una serie de posibles funcionalidades no contempladas que pueden enriquecer las posibilidades de comunicación entre el usuario y la aplicación (por ejemplo: opción para recuperar la contraseña, envíos de emails informativos, recordar el inicio de sesión,).

Por otro lado, se deja abierta la posibilidad de poder incluir información de cualquier otra zona que no sea la Sierra de las Nieves, de forma que se pueda trasladar a cualquier escenario, lo que sin duda puede ayudar a alcanzar una mayor difusión y una posible viabilidad y rentabilidad económica. En cualquier caso, el desarrollo del proyecto ha tenido en cuenta otros escenarios, y se ha diseñado de forma que sea fácil incorporar otras zonas.

Así mismo, el siguiente paso será añadir el algoritmo de cálculo de rutas recomendadas desarrollado por el otro proyecto vinculado a éste.

ANEXO I: MANUAL DE INSTALACIÓN

CAPÍTULO

18 Instalación del servidor

En cuanto a los recursos software empleados para el proyecto caben destacar los siguientes:

- Sistema Operativo Linux Ubuntu 10.04.
- Editores de texto Gedit, Kompozer y Geany como entorno de desarrollo para la creación y edición del código fuente.
- Servidor web Apache 2, bajo licencia Apache License.
- Servidor de mapas UMN MapServer, bajo licencia X/MIT.
- Gestor de bases de datos PostgreSQL 8.4 (bajo licencia BSD) con las extensiones PostGIS 1.5.1 (licencia GPL) y pgrouting 1.01 (licencia GPL v2). Además, para facilitar las tareas de administración se han usado los entornos gráficos pgAdmin 3 y phpPgAdmin (web).
- Aplicación SIG de escritorio gvSIG, bajo licencia GPL v2.
- Librería OpenLayers, bajo licencia FreeBSD.
- Herramienta para creación de diagramas de flujo DIA, bajo licencia GPL.
- Editor de imágenes Gimp 2, para los gráficos presentes en la página web, bajo licencia GPL.

Sistema Operativo

El Sistema Operativo utilizado ha sido Linux Ubuntu 10.04, de la rama Debian, una distribución que goza de un gran soporte por parte de la comunidad de software libre.

Este Sistema Operativo es libre y gratuito, y se puede descargar desde la página oficial (www.ubuntu.com).

Servidor web

Para dotar al servidor de la capacidad de atender y dispensar páginas web mediante el protocolo estándar HTTP, se ha instalado Apache 2, el servidor más popular usado en Internet. La aplicación se encuentra en los repositorios oficiales de Ubuntu, y para instalarla hay que introducir el siguiente comando en un terminal:

```
sudo apt-get install apache2
```

Además, dado que la aplicación web se basará en contenido dinámico, se optará por usar el módulo de PHP como lenguaje del lado del servidor para dotar a Apache de la capacidad de tratar con contenido dinámico. Para ello hay instalar el modulo PHP para Apache. Tras terminar la instalación, hay que reiniciar el servidor web.

```
sudo apt-get install php5 libapache2-mod-php5
sudo /etc/init.d/apache2 restart
```

Base de Datos

El siguiente paso es instalar la base de datos PostgreSQL (versión 8.4), junto con la extensión PostGIS y la librería pgRouting. Éste no se encuentra en los repositorios oficiales de Ubuntu, pero existe una comunidad llamada UbuntuGIS que ofrece unos repositorios actualizados de software geoespacial, entre las que se encuentra pgRouting.

Por tanto, lo primero que hay que hacer es añadir el repositorio en Ubuntu, con los siguientes comandos:

```
sudo add-apt-repository ppa:georepublic/pgrouting
sudo apt-get update
sudo add-apt-repository ppa:ubuntugis/ppa
sudo add-apt-repository ppa:ubuntugis/ppa
```

Y después proceder a la instalación conjunta de PostgreSQL, PostGIS y pgRouting:

```
sudo apt-get install gaul-devel \
        postgresql-8.4-pgrouting \
        postgresql-8.4-pgrouting-dd \
        postgresql-8.4-pgrouting-tsp
```

El gestor de base de datos quedará instalado, aunque el modo de acceder e interactuar con él será a través de la línea de comandos del terminal. Si se desea utilizar una interfaz gráfica, el centro de software de Ubuntu ofrece la aplicación pgAdmin3. Si se desea una interfaz web, se puede instalar phpPgAdmin:

```
sudo apt-get install phppgadmin
```

Y a continuación reiniciar Apache:

```
sudo /etc/init.d/apache2 restart
```

Comunicando la aplicación con la base de datos

Cuando la aplicación web necesite dialogar con la base de datos utilizará el lenguaje PHP, y para ello se instalará el modulo de PHP para PostgreSQL. Tras ello, reiniciar el servidor web:

```
sudo apt-get install php5-pgsql
sudo /etc/init.d/apache2 restart
```

Servidor de mapas. UMN MapServer.

Un servidor de mapas de Internet (IMS las siglas en inglés) es un servicio que provee mapas a través de la red, normalmente como imágenes. Una especificación estándar para este tipo de servidores es el OGC Web Map Service (WMS).

```
sudo apt-get install mapserver
sudo /etc/init.d/apache2 restart
```

Esto instalará también las librerías espaciales GDAL/OGR y PROJ.4.

Para comprobar que MapServer funciona correctamente, introducir en el navegador la siguiente dirección:

http://localhost/cgi-bin/mapserv

CAPÍTULO

19 Instalación de la aplicación

Crear una base de datos geoespacial

Para crear una base de datos geoespacial hay que hacer lo siguiente:

```
sudo su postgres
# crea base de datos
createdb routing


#PostGIS precisa del lenguaje PL/pgSQL para utilizar sus funciones
createlang plpgsql routing


# funciones PostGIS
psql -d routing -f /usr/share/postgresql/8.4/contrib/postgis-1.5/
postgis.sql
psql -d routing -f /usr/share/postgresql/8.4/contrib/postgis-1.5/
spatial_ref_sys.sql


# funciones pgRouting
psql -d routing -f /usr/share/postlbs/routing_core.sql
psql -d routing -f /usr/share/postlbs/routing_core_wrappers.sql
psql -d routing -f /usr/share/postlbs/routing_topology.sql
```

Con esto ya tenemos una base de datos geoespacial con todas las funciones de PostGIS y pgRouting disponibles.

Cambio del sistema de coordenadas y proyección

El datum ED50 (European Datum 1950) fue concebido tras la Segunda Guerra Mundial como el sistema geodésico de referencia oficial para dotar a toda Europa de una homegeneidad geodésica y cartográfica, presentando desviaciones máximas de 10 metros. Este sistema no fue acogido como oficial por todos los países, y a medida que se hacía patente la necesidad de unificar la cartografía mundial con la creación de sistemas globales de navegación por satélite se decidió adoptar en 1989 un nuevo sistema de referencia, el ETRS89, más exacto que el ED50 y consistente con los sistemas globales de geoposicionamiento, lo que ayudó a que esta vez sí fuera adoptado por la mayoría de países europeos.

El 27 de julio de 2007 se publicó en el BOE (Boletín Oficial del Estado) el Real Decreto 1071/2007, por el que se regulaba el sistema geodésico de referencia oficial en España, donde se adoptó el sistema ETRS89 (European Terrestrial Referencia System 1989) como nuevo sistema de referencia geodésico oficial en España, sustituyendo al sistema geodésico de referencia regional ED50, el oficial hasta ese momento en el país, y dando un plazo de adaptación a la norma hasta 2015 para realizar la transición de un sistema a otro.

Con este panorama, se ha optado por seguir esta nueva regulación en el presente proyecto y disponer de todos los datos geoespaciales en el sistema ETRS89.

Una ventaja adicional de adoptar esta decisión es que el datum de ETRS89 es el mismo que el de WGS84, el sistema geodésico de referencia que usan los navegadores GPS y los archivos KML (formato elegido para la descarga de rutas de la aplicación web), por lo que a la hora de generar un KML sólo habrá que reproyectar de UTM a coordenadas geográficas, ahorrando tiempo de cálculo a la base de datos PostGIS.

Los datos de partida proporcionados por el cliente se encuentran en formato ED50, por lo que antes de incluirlos en la IDE se ha optado por transformarlos a ETRS89. Adoptar esta decisión ahorra tiempo de cálculo al servidor de mapas MapServer, dado que estar reproyectando de ED50 a ETRS89 en cada petición consume tiempo de ejecución, mientras que una única transformación en el origen de los datos libera a MapServer de todo ese posterior trabajo.

El proceso de transformación de ED50 a ETRS89 no es inmediato. Unas coordenadas para el datum ETRS89 distan de unas del datum ED50 en casi 200 metros, así que es recomendable usar una rejilla de transformación de datum que contenga los incrementos en longitud y latitud entre esos dos datums. Esta transformación no destruye topología alguna, es eficiente y continua con un grado de acuerdo de unos pocos centímetros sobre todo el territorio. El Instituto Nacional de Geografía proporciona dos rejillas, una para la Península Ibérica y otra para Baleares, para realizar esta transformación.

Las librerías GDAL/OGR tienen una herramienta llamada ogr2ogr que puede realizar este tipo de conversiones. Tan sólo hay que indicar los datos de la rejilla (proporcionados por el IGN) en el parámetro +towgs84 a la hora de especificar los datos de partida (ED50 huso 30N, o lo que es lo mismo, EPSG:23030). Por ejemplo, si se quisiera transformar un archivo llamado "rutas.shp", que está en ED50 huso 30N (EPSG: 23030), a ETRS89 huso 30N (EPSG:25380), habría que hacer lo siguiente:

```
ogr2ogr -s_srs "+init=epsg:23030 +units=m  +towgs84=-131, -100.3, -163.4,
-1.244,-0.020,-1.144,9.39 +wktext" -t_srs "+init=epsg:25830 +nadgrids
+wktext" rutas_25830 rutas.shp
```

Esto crea un directorio "rutas_25830" con "rutas.shp" en el sistema EPSG:25830 y varios archivos, entre ellos uno con extensión ".proj" que contiene información del nuevo sistema de proyección y referencia. Con esta transformación del datum a ETRS89, la conversión a WGS84 (EPSG:4326) no necesita de ninguna rejilla, dado que ambos datums coinciden, como ya se ha comentado:

```
ogr2ogr -s_srs EPSG:25830 -t_srs EPSG:4326 rutas_4326.shp rutas.shp
```

Este comando genera un archivo "rutas_4326.shp" con sistema de proyección y coordenadas EPSG:4326 (datum WGS84, coordenadas en grados -longitud y latitud-).

Cargar datos geoespaciales en la base de datos

El primer paso es ver qué tipo de datos geoespaciales disponemos. En el proyecto se proporcionaron datos en formato Shapefile.

El paquete que instaló PostgreSQL incorpora una herramienta llamada Shp2pgsql que convierte Shapefiles en sentencias SQL listas para ser insertadas en una base de datos de PostgreSQL/PostGIS tanto en formato geométrico como geográfico. Al ejecutarse el script SQL en la base de datos, se creará una tabla con los datos geoespaciales.

El comando que hay introducir es el siguiente:

```
shp2pgsql -s 4326 -W LATIN1 viariopn.shp public.viariopn > viario.sql
```

Esto convierte "viariopn.shp" en un fichero "viario.sql" listo para ser ejecutado en SQL en la base de datos, que creará una tabla de nombre "viariopn" (en el esquema "public"). La opción "-s 4326" indica el EPSG de los datos de partida (en este caso suponemos que es EPSG:4326), mientras que "-W LATIN1" indica la codificación del Shapefile (el cliente entregó los datos en este formato, que acepta caracteres latinos, como la eñe).

Si todo ha ido bien, responderá indicando el tipo de datos incluidos en el Shapefile:

Shapefile type: Arc

Postgis type: MULTILINESTRING[2]

Una vez obtenido el archivo "viario.sql", se ejecuta sobre la base de datos (cuyo nombre es "routing"):

```
psql -U postgres -d routing -f viario.sql
```

Calcular la topología: construyendo el grafo

Para trabajar con las rutas y los algoritmos tanto de calculo de ruta más corta como de ruta recomendada (multicriterio), es necesario dotar a los datos de las rutas de una topología en forma de grafo, con nodos y arcos. La extensión pgRouting proporciona un método para poder calcular dicha topología, llamada *assign_vertex_id()*. La función asignará los vértices origen y destino de un segmento, así que antes de ejecutar la función hay que crear dos nuevos atributos en la tabla "viariopn" que indicarán precisamente esos dos nodos:

```
ALTER TABLE public.viariopn ADD COLUMN "source" integer;
ALTER TABLE public.viariopn ADD COLUMN "target" integer;
```

Ya se puede ejecutar la función que asigna vértices:

```
SELECT assign_vertex_id('viariopn', 0.01, 'the_geom', 'gid');
```

El segundo parámetro es la tolerancia, que determina el margen para tomar como vértice una discontinuidad en un vector o línea cuya distancia sea menor que el valor del parámetro de tolerancia. Éste será mayor o menor según la calidad de los datos suministrados. hay que tener en cuenta el sistema de referencia que se usa, ya que la tolerancia puede estar en metros o en grados.

Para incluir los otros algoritmos hay que hacer lo siguiente:

```
--A Star

ALTER TABLE viario_pn ADD COLUMN x1 double precision;

ALTER TABLE viario_pn ADD COLUMN y1 double precision;

ALTER TABLE viario_pn ADD COLUMN x2 double precision;

ALTER TABLE viario_pn ADD COLUMN y2 double precision;


UPDATE viario_pn SET x1 = x(ST_startpoint(the_geom));

UPDATE viario_pn SET y1 = y(ST_startpoint(the_geom));


UPDATE viario_pn SET x2 = x(ST_endpoint(the_geom));

UPDATE viario_pn SET y2 = y(ST_endpoint(the_geom));


UPDATE viario_pn SET x1 = x(ST_PointN(the_geom, 1));

UPDATE viario_pn SET y1 = y(ST_PointN(the_geom, 1));


UPDATE viario_pn SET x2 = x(ST_PointN(the_geom, ST_NumPoints(the_geom)));

UPDATE viario_pn SET y2 = y(ST_PointN(the_geom, ST_NumPoints(the_geom)));




--Shooting-Star

-- Add rule and to_cost column

ALTER TABLE viario_pn ADD COLUMN to_cost double precision;

ALTER TABLE viario_pn ADD COLUMN rule text;
```

La elección de la tolerancia es uno de los puntos más delicados de todo el proceso, ya que el parámetro de tolerancia debe ser escogido teniendo en cuenta la calidad de los datos proporcionados por el cliente (en el shp), es decir la precisión con la que fueron tomados en su día con un aparato GPS. Es común encontrar intersecciones donde las rutas no llegan a conectar, o simplemente secciones de una ruta inconexas. La tolerancia trata en este sentido de eliminar esas inconexiones no intencionadas, y su valor indicará que en un radio de x metros (o grados, según el sistema de referencia, otra complicación más) todos los puntos de inicio o final de un segmento serán tratados como un solo nodo.

Desgraciadamente, no existe una función en PostGIS o pgRouting que estudie un conjunto de datos y nos indique el valor óptimo de tolerancia para aplicar a dicho conjunto, así que no queda más remedio que usar el método de prueba y error. Pero tampoco disponemos de una función que nos diga si la topología creada está llena de inconexiones, y en realidad no tendría mucho sentido, ya que pueden existir "inconexiones" válidas y que deben estar ahí, como un sendero que termina en una cima o una calle sin salida. Así que una forma de afrontar el problema es visualizar de alguna forma esas inconexiones, y pgRouting nos proporciona de forma indirecta la solución: al crear la topología, crea una tabla temporal llamada *vertices_tmp* donde se encuentran todos los nodos creados. Creando un atributo llamado "cnt", que cuente para cada nodo cuántas entradas con ese número de nodo hay en el viario, podremos averiguar los puntos inconexos (serán aquellos cuyo valor "cnt" sea igual a 1).

Incluyendo la siguiente capa en MapServer, podemos visualizar esas inconexiones:

Capa de puntos muertos (deadends) MapServer

```
LAYER
OFFSITE 255 255 255
#LABELITEM 'name'
TOLERANCE 20
NAME            'dead_ends'
TYPE            POINT
STATUS  DEFAULT
CONNECTIONTYPE postgis
CONNECTION 'user=user password=user dbname=routing6 host=localhost port=5432'
DATA 'the_geom from vertices_tmp as foo using unique id using SRID=4326'

CLASSITEM 'cnt'

METADATA
'WMS_TITLE'   'dead_ends'
'WMS_SRS'     "EPSG:4326"
'WMS_INCLUDE_ITEMS' 'all'
END
CLASS
Text ([id])
```

```
EXPRESSION /1/
STYLE
SYMBOL 'tent'
SIZE 11
COLOR 255 0 0
END

LABEL
    TYPE TRUETYPE
    ANTIALIAS TRUE
    FONT 'vera'
    COLOR 255 0 0
    BACKGROUNDCOLOR 240 240 240
     POSITION cr
      MINSIZE 8
    MAXSIZE 12
    BUFFER 2
     END
END

CLASS
TEXT ([id])
EXPRESSION /./

STYLE
SYMBOL 'tent'
SIZE 11
COLOR 0 0 255
END

LABEL
    TYPE TRUETYPE
    ANTIALIAS TRUE
    FONT 'vera'
    COLOR 0 0 255
    BACKGROUNDCOLOR 240 240 240
```

```
        POSITION cr
         MINSIZE 8
        MAXSIZE 12
        BUFFER 2
         END
END
END
END
```

Aquí se puede apreciar los puntos inconexos, en rojo:

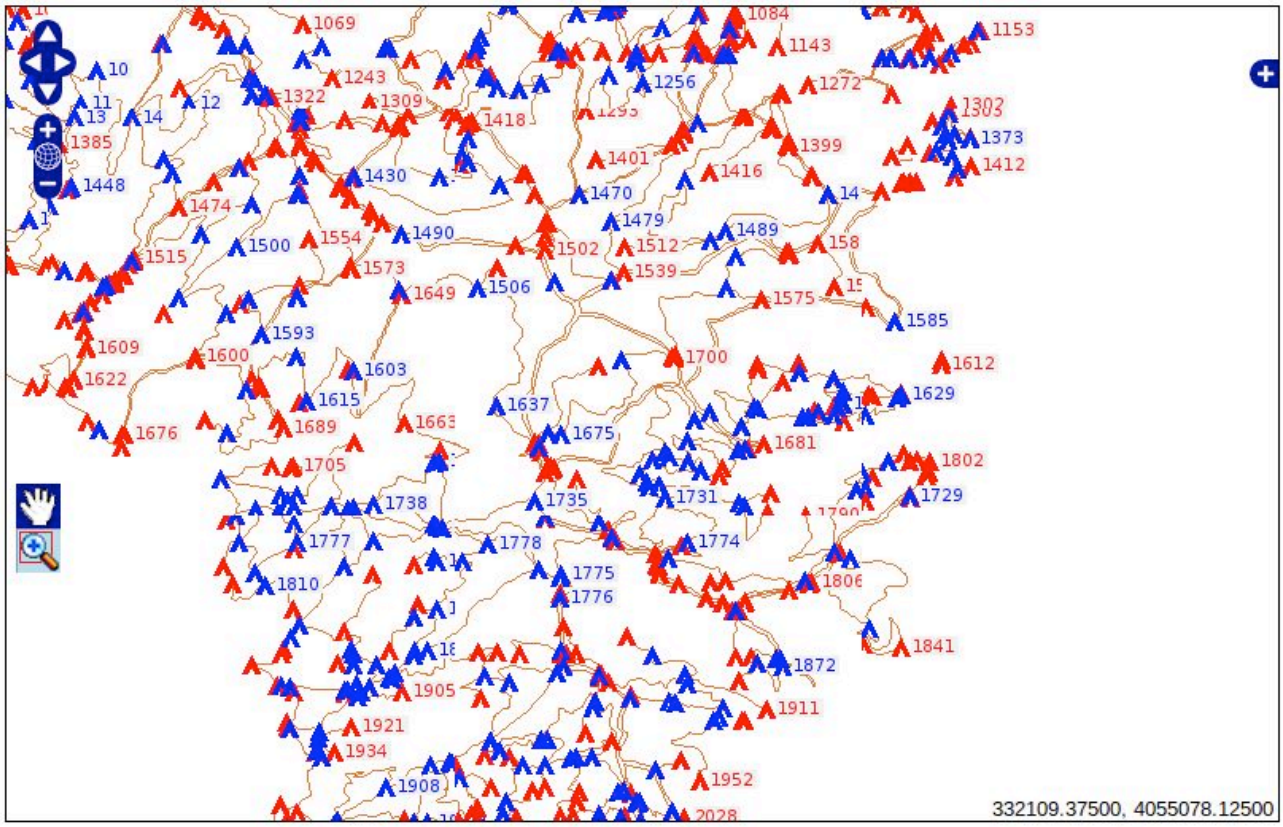

Figura A-I.1. Mapa con inconexiones (I).

Si hacemos zoom sobre una zona, apreciamos con más detalle dónde se producen:

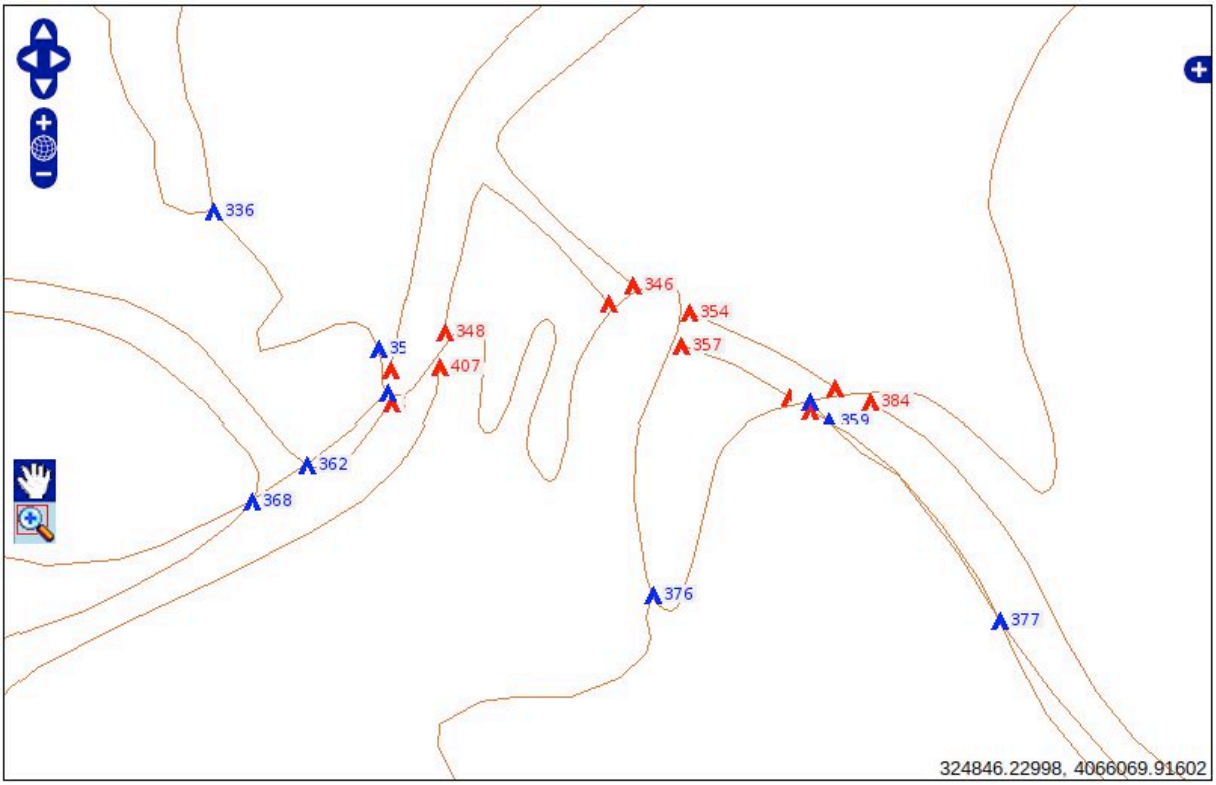

Figura A-I.2. Mapa con inconexiones (II).

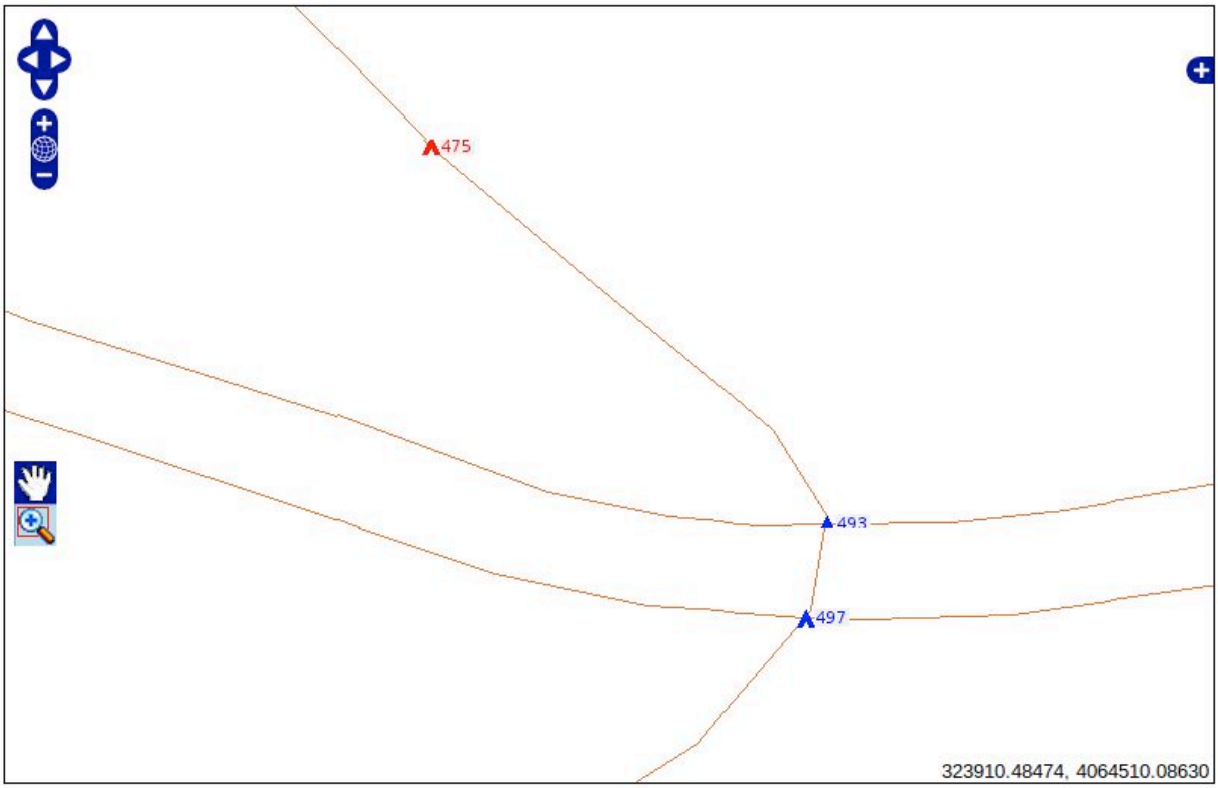

Figura A-I.3. Mapa con inconexiones (III).

Una cantidad excesiva de inconexiones provocará una funcionamiento deficiente del cálculo de rutas:

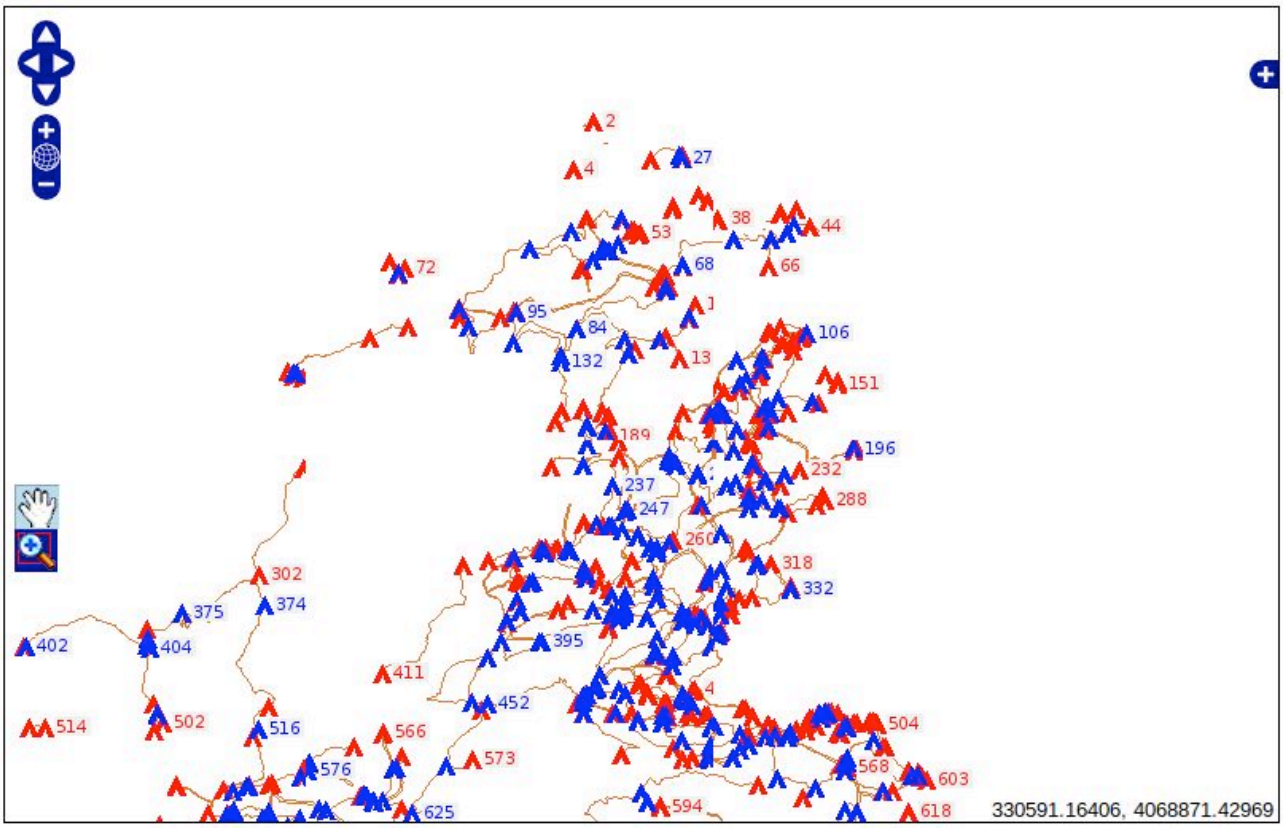

Figura A-I.4. Mapa con inconexiones (IV).

Gestión de usuarios

La gestión de usuarios se va a llevar a cabo a través de la base de datos, por lo que hay que crear una tabla que almacene los datos de las cuentas de usuario.

```
--Crear tabla usuarios
CREATE TABLE usuarios (user_id integer NOT NULL,
    username character varying(11) NOT NULL,
    password character varying(32) NOT NULL,
    email character varying(40) NOT NULL,
    perfil integer
);
CREATE SEQUENCE usuarios_user_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER SEQUENCE usuarios_user_id_seq OWNED BY usuarios.user_id;



ALTER TABLE usuarios ALTER COLUMN user_id SET DEFAULT nextval
('usuarios_user_id_seq'::regclass);
```

Adicionalmente, también hay que incluir una tabla que almacena las rutas guardadas por los usuarios.

```
--Tablas de rutas de usuarios (senderos_usuarios)
CREATE TABLE senderos_usuarios (
    id integer NOT NULL,
    usuario character varying(11) NOT NULL,
    gid integer NOT NULL,
    nombre character varying(35),
    length integer,
    perfil character varying(1),
    dificultad character varying(10),
    horas double precision,
    fecha date,
    motivacion character varying(50),
    the_geom geometry,
    CONSTRAINT enforce_dims_the_geom CHECK ((st_ndims(the_geom) = 2)),
     CONSTRAINT enforce_geotype_the_geom CHECK (((geometrytype(the_geom) =
'MULTILINESTRING'::text) OR (the_geom IS NULL))),
    CONSTRAINT enforce_srid_the_geom CHECK ((st_srid(the_geom) = 4326))
);
```

Estructura en directorios

Como se ha comentado en capítulso anteriores, la aplicación quedará estructurada en lo siguientes directorios, que se incluyen en el CD que acompaña a la memoria.

css/

images/

includes/

js/

layouts/

modulos/

OpenLayers/

ANEXO II: MANUAL DE CÓDIGO

Manual de código

En esta sección se muestra el código o parte del código de algunos de los archivos más importantes de la aplicación que son mencionados a lo largo de la memoria, por si el lector quiere profundizar más en ellos.

Mapfile

```
MAP
   NAME mapasnieves
   IMAGETYPE      png
   SYMBOLSET "mapa23.sym"
   FONTSET "fonts.txt"
   SHAPEPATH "/home/user/shp/"
   CONFIG "PROJ_LIB" "/home/user"
   LEGEND
      IMAGECOLOR -1 -1 -1
      LABEL
         FONT "vera"
         ANGLE FOLLOW
         COLOR 0 0 0
         ENCODING "UTF-8"
         TYPE truetype
         SIZE 8
      END
      STATUS ON
      TRANSPARENT ON
   END
   WEB
      METADATA
         "wms_title" "Mapserver Sierra de las Nieves"
         "wms_srs" "EPSG:4326 EPSG:900913"
         "wms_onlineresource" "http://localhost/cgi-bin/mapserv?map=/
home/user/01aptwms.map"
      END
   END
   PROJECTION
      "init=epsg:900913"
   END
   OUTPUTFORMAT
         NAME png
         DRIVER "GD/PNG"
         MIMETYPE "image/png"
         IMAGEMODE RGBA
         EXTENSION "png"
          TRANSPARENT ON
   END

   LAYER
      NAME "viariopn5"
      STATUS ON
      TYPE LINE
      DATA "viario_pn_5_4326.shp"
      MAXSCALE -1.0
      MINSCALE -1.0

      SIZEUNITS pixels
      PROJECTION
         "init=epsg:4326"
      END
      CLASS
         STYLE
            COLOR 210 105 30
            WIDTH 2
         END
         NAME "default"
      END
```

```
METADATA
         "wms_title" "viariopn5"
       "wms_srs" "EPSG:4326 EPSG:900913"
         "wms_abstract" "generated by gvSIG"
         "gml_include_items" "all"
      END
   END # Layer

  LAYER
      NAME "hitos"
      STATUS ON
      TYPE POINT
      OFFSITE 255 255 255
      TOLERANCE 20
      DATA "hitos_4326.shp"
      MAXSCALE -1.0
      MINSCALE -1.0
      SIZEUNITS pixels
      PROJECTION
         "init=epsg:4326"
      END
      CLASS
         STYLE
            COLOR 100 149 237
            WIDTH 3
         END
         NAME "default"
      END
      METADATA
         "wms_title" "hitos"
       "wms_srs" "EPSG:4326 EPSG:900913"
         "wms_abstract" "generated by gvSIG"
         "gml_include_items" "all"
      END
   END # Layer
END # Map File
```

Archivo index.php

```
<?php
session_start(); // primero la sesión para evitar warnings
error_reporting(E_ALL);
ini_set('display_errors', 1);

//Si es usuario anonimo
if (!isset($_SESSION['SESSION_UNAME']))
{
     $tipousuario='anon';
}
else if (isset($_SESSION['SESSION_UADMIN']))
{
     $tipousuario='admin';
}
else
{
     $tipousuario='reg';
}

// Primero incluimos el archivo de configuración
include('conf.php');

/** Verificamos que se haya escogido un modulo, si no
* tomamos el valor por defecto de la configuración.
*/
if (!empty($_GET['mod']))
     $modulo = $_GET['mod'];
else
     $modulo = MODULO_DEFECTO;

if (empty($conf[$modulo]))
           $modulo = MODULO_DEFECTO;


/*Comprobamos si el usuario tiene permiso para ver ese modulo*/
//Si no tiene permiso lanzo mensaje de error
if (($conf[$modulo]['tipousuario']!='todos') && ($conf[$modulo]['tipousuario']!=
$tipousuario) && ($tipousuario!='admin'))
{
     header("Location: index.php?mod=error&e=2");
     exit();
}


/** archivo de Layout (plantilla) */
$path_layout = LAYOUT_PATH.'/'.$conf[$modulo]['layout'];
$path_modulo = MODULO_PATH.'/'.$conf[$modulo]['archivo'];
$path_head = HEAD_PATH.'/'.$conf[$modulo]['head'];
$titulo = $conf[$modulo]['titulo'];

if (file_exists($path_layout))
     include( $path_layout );
else
     if (file_exists( $path_modulo ))
         include( $path_modulo );
     else
          die('Error al cargar el módulo <b>'.$modulo.'</b>. No existe el archivo
<b>'.$conf[$modulo]['archivo'].'</b>');
?>
```

Archivo conf.php (parcial)

```
<?php
/*
 * Archivo de configuración para nuestra aplicación modularizada.
 * Definimos valores por defecto y datos para cada uno de nuestros módulos.
*/
define('MODULO_DEFECTO', 'intro');
define('LAYOUT_DEFECTO', 'layout_2.php');
define('HEAD_DEFECTO', 'head.html');
define('MODULO_PATH', realpath('./modulos/'));
define('LAYOUT_PATH', realpath('./layouts/'));
define('HEAD_PATH', realpath('./layouts/'));

$conf['intro'] = array(
           'archivo' => 'intro.html',
           'layout' => 'layout_intro.php',
                'head' => 'headintro.html',
           'tipousuario' => 'todos',
              'titulo' => 'Sierra de las Nieves :: Web de rutas senderistas');
$conf['home'] = array(
           'archivo' => 'home.php',
           'layout' => LAYOUT_DEFECTO,
                'head' => 'headhome.html',
           'tipousuario' => 'todos',
              'titulo' => 'Sierra de las Nieves :: Inicio');
$conf['registro'] = array(
           'archivo' => 'registro.php',
           'layout' => LAYOUT_DEFECTO,
                'head' => 'headreg.html',
           'tipousuario' => 'anon',
              'titulo' => 'Sierra de las Nieves :: Registro de nuevo usuario');
$conf['buscarutas'] = array(
           'archivo' => 'buscarutas.php',
           'layout' => LAYOUT_DEFECTO,
                'head' => 'headbusca.html',
           'tipousuario' => 'todos',
              'titulo' => 'Sierra de las Nieves :: Buscador de rutas');
$conf['calculoapt'] = array(
           'archivo' => 'apt.php',
           'layout' => LAYOUT_DEFECTO,
                'head' => 'headapt.html',
           'tipousuario' => 'todos',
              'titulo' => 'Sierra de las Nieves :: Cálculo de ruta por aptitudes');
$conf['calculoaptres'] = array(
           'archivo' => 'aptres.php',
           'layout' => 'layout_2mapa.php',
                'head' => 'head_mapares.php',
           'tipousuario' => 'todos',
              'titulo' => 'Sierra de las Nieves :: Rutas resultado por aptitudes');
$conf['calculodij'] = array(
           'archivo' => 'calcdij.php',
           'layout' => 'layout_2mapa.php',
                'head' => 'head_mapadij.html',
           'tipousuario' => 'todos',
              'titulo' => 'Sierra de las Nieves :: Cálculo de la ruta más corta');

/*etc etc*/

?>
```

Archivo loadmapres.php (parcial, comunicación AJAX)

```
function agregaruta(i,id)
     {
          cargaruta=parseInt(i);
          //XMLHttpRequest a agregaruta.php
          var xmlhttp=new XMLHttpRequest();
          xmlhttp.onreadystatechange=function()
          {
               if (xmlhttp.readyState==4 && xmlhttp.status==200)
                  {
                    if (xmlhttp.responseText=='si')
                    {
                         apprise('Sendero agregado a tu lista de rutas',
{'animate':true});
                         //cambiar icono por uno gris y quitar addeventlistener
(para que no añada otra vez la misma)
                         var boton = document.getElementById(id);
                         boton.setAttribute("class", "botonAgregarDes");
                         boton.removeEventListener('click',fagregar, false);
                    }
                    else
                    {
                         apprise('Error, no se pudo agregar a tu lista de rutas',
{'animate':true});
                    }
                  }
          }
          xmlhttp.open("GET","agregaruta.php?gid="+parseInt(i),true);
          xmlhttp.send();

     }
```

Glosario de términos

En este apéndice se pretende aglutinar todos los términos relacionados directamente con este proyecto que aparecen habitualmente en esta memoria, son definidos en alguna parte pero para facilitar búsqueda está este glosario:

• Sistema de Información Geográfica (SIG, GIS las siglas en inglés): sistema que auna tanto capacidades de cálculo geográfico (el llamado procesamiento geográfico) como de manipulación gráfica y de imágenes, consulta y gestión de bases de datos.

• Infraestructura de Datos Espaciales (IDE): conjunto de tecnologías que facilitan la integración, la disponibilidad y el acceso a la información espacial en Internet. Existen varias definiciones sobre qué es una IDE, pero todas destacan los siguientes puntos: una IDE debe estar formada por conjuntos de datos espaciales. servicios de datos geográficos espaciales y metadatos (que serán el índice que describa los datos y los servicios), junto con las tecnologías de red necesarias para proporcionar acceso a esos datos.

• Open Geospatial Consortium (OGC): consorcio encargado de la estandarización de los datos y servicios que involucran información geoespacial.

• Open Source Geospatial Foundation (OSGeo): organización que reúne y provee soporte al software geoespacial de código abierto.

• PostgreSQL: sistema de gestión de base de datos relacional, publicado bajo licencia de software libre BSD.

• PostGIS: módulo para PostgreSQL desarrollado principalmente por Refractions Research Inc., con el cual PostgreSQL adquiere la capacidad de almacenar información geoespacial (cumpliendo el estándar SFSS, certificado en 2006 por el OGC, lo que garantiza la interoperabilidad con otros sistemas estándar de información geográfica) y de realizar operaciones de análisis geográfico.

• PgRouting: extensión para PostgreSQL+PostGIS que provee un potente motor de cálculo de rutas y otras operaciones geoespaciales.

• Servidor de mapas de Internet: es un servicio que provee mapas a traves de Internet, normalmente como imágenes.

• UMN MapServer: es un servidor de mapas de código abierto, una plataforma para publicación de datos espaciales y aplicación con mapas interactivos en la web. Pertenece al consorcio OSGeo.

• Mapfile: archivo de configuración del servidor de mapas MapServer.

• Web Map Service (WMS): protocolo/especificación estándar internacional del OGC para visualizar mapas de datos geoespaciales de forma dinámica a partir de información geográfica en formato de imagen (como PNG, GIF o JPEG), o vectorial (SVG). Define tres operaciones: devolver metadatos del nivel de servicio; devovler un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos; devolver información de características particulares mostradas en el mapa (opcionales).

• Proj.4: librería que contiene información sobre los sistemas de referencia EPSG y CRC.

• Shapefile: formato de archivo de datos espaciales de ESRI (Enviromental Systems Research Institute).

• Keyhole Markup Language (KML): formato popularizado por Google Earth que está basado en XML, y sirve para almacenar y representar datos geográficos.

• European Petroleum Survey Group (EPSG): organismo que entre otras cosas es la responsable de la tabla de códigos relativo a sistemas de referencia (sistemas de coordenadas, proyecciones cartográficas...) utilizados en la representación cartográfica, que facilita la definición de dichos sistemas en un entorno informático. Por ejemplo, EPSG:23030 es el código de la proyección UTM ED50 en el huso 30N.

• Servidor web (servidor HTTP): es un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. Un servidor web popular es Apache.

• OpenLayers: librería JavasScript para mostrar mapas de forma dinámica en navegadores web, sin depender del lado del servidor. Es un proyecto del OSGeo, y lo usan webs como OpenStreetMaps.

• HTML: lenguaje de programación web que define la estructura de las páginas web.

• CSS: lenguaje de programación web que define el aspecto de los documentos HTML.

• PHP: lenguaje de programación web ejecutado por el servidor.

• JavaScript: lenguaje de programación web que define el comportamiento de las páginas web.

• jQuery: conjunto de librerías escritas en JavaScript.

Bibliografía

[1]: SANTOS PRECIADO, JOSÉ MIGUEL (2004): "Sistemas de Información Geográfica". Ed. UNED, Madrid.

[2]: GÓMEZ DELGADO, MONTSERRAT y BARREDO CANO, JOSÉ I. (2005): "Sistemas de Información Geográfica y evaluación multicriterio en la ordenación del territorio". RAMA, Madrid.

[3]: BOSQUE SENDRA, JOAQUÍN Y GARCÍA, ROSA C. (2000): “El uso de los sistemas de información geográfica en la planificación territorial”. Universidad Complutense.

[4]: CAPDEVILLA I SUBIRANA, JOAN (2004): "Infraestructura de Datos Espaciales (IDE). Definición y desarrollo actual en España". Art. revista Script Nova. Vol. VIII, núm. 170. Universidad de Barcelona.

[5]: GONZALEZ, PEDRO A.; LORENZO, MIGUEL; LUACES, MIGUEL R. y PARAMÁ, JOSÉ R. (2006): "Un nodo local de la IDE de España: ideAC". Art. Universidad de La Coruña.

[6]: NIELSEN, JAKOB Y LORANGER, HOA (2006): “Usabilidad, prioridad en el diseño web”. Anaya Multimedia.

[7]: Consorcio W3C: www.w3c.es.

[8]: Consorcio OGC: www.opengeospatial.org.

[9]: OSGeo: www.osgeo.org.

[10]: FRANCO REY, JORGE (2002): “Nociones de Cartografía”. Cartesia.

[11]: EPSG: www.epsg.org.

[12]: BOOCH, GRADY; RUMBBAUGH, JAMES; JACOBSON, IVAR J. (2009): “UML: El lenguaje unificado de modelado: guía de usuario.”. Addison Wesley.

[13]: Apache Software Foundation: www.apache.org.

[14]: UMN MapServer: www.mapserver.org.

[15]: PostgreSQL: www.postgresql.org.

[16]: PostGIS: postgis.refractions.net.

[17]: PgRouting: www.pgrouting.org.

[18]: OpenLayers: www.openlayers.org.

[19]: ESRI Shapefile: www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

[20]: GeoTIFF: geotiff.osgeo.org.

[21]: GeoJSON: www.geojson.org.

[22]: KML: code.google.com/apis/kml/documentation/.

[23]: Modelo de las tres capas: http://www.cristalab.com/tutoriales/javascript-no-intrusivo-css-y-php-c31507l/

[24]: Diseño modular: http://www.zonaphp.com/creando-webs-modulares/

Otras obras consultadas:

GÓMEZ CASTAÑO, JOSÉ (2010) : "Desarrollo de una IDE ferroviaria basada en software libre". Art. IV Jornadas de SIG libre, Gerona.

SANZ SALINA, J. GASPAR y MONTESINOS LAJARA, MIGUEL (2006): "Panorama actual del ecosistema de SIG libre". Art. Prodevelop.

JOLY, FERNAND (1982): "La cartografía". Ariel, Barcelona.

BASELGA MORENO, SERGIO (2006): "Fundamentos de cartografía matemática". Universidad Politécnica de Valencia.

COMAS, DAVID y RUIZ, ERNESTO (1993): "Fundamentos de los Sistemas de Información Geográfica". Edit. Ariel, Barcelona.

LAURINI, ROBERT y THOMPSON, DEREK (1992): "Fundamentals of Spatial Information Systems". Academic Press, London.

LUQUE RUIZ, I. y GÓMEZ-NIETO, M.A. (1997): "Diseño y uso de bases de datos relacionales". Ediciones Ra-Ma Madrid.

SAMET, HANAN (1990): "Applications of Spatial Data Structures". Addison-Wesley Publishing Company, New York.

Índice de figuras